A DUAL FINITE ELEMENT COMPUTER SYSTEM FOR STRETCHING AND BENDING OF ORTHOTROPIC PLATES

by

Dennis Alan Nagy

Sponsored by
The National Aeronautics and Space Administration
Research Grant NRG-22-009-059

N67-37882 (ACCESSION NUMBER) (THRU) 1
209 (PAGES) (CODE) 32
CR# 88866 (NASA CR OR TMX OR AD NUMBER) (CATEGORY)
FACILITY FORM 602

MIT

DEPARTMENT OF CIVIL ENGINEERING

SCHOOL OF ENGINEERING
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge 39, Massachusetts

R76-44

September 1967

A DUAL FINITE ELEMENT COMPUTER SYSTEM

FOR

STRETCHING AND BENDING OF ORTHOTROPIC PLATES

by

DENNIS ALAN NAGY*

September 1967

* Graduate student, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts.

FOREWORD

The work described herein was performed at the Civil Engineering Department, Massachusetts Institute of Technology under the supervision of Professor Z. M. Elias. It served for the author as a partial fulfillment of the requirements for the degree of Master of Science. It was supported by Research Grant NGR-22-009-059-(251) from the National Aeronautics and Space Administration.

ABSTRACT

The objects of this report are 1) a study of the dual displacement-stress function finite element method in matrix form, with particular consideration of a number of different types of boundary conditions, and 2) implementation of the method as a computer system for the solution of both plate stretching and plate bending problems. The computer system is developed as an addition to the general capabilities of the Finite Element Analyzer, which in turn is a part of the problem-oriented Structural Design Language (STRUDL) within the Integrated Civil Engineering System (ICES).

The finite element displacement method as applied to stretching of triangular plate elements is presented with particular reference to formulation of the system equations in matrix form. The dual stress function method for bending of triangular plate elements is then presented in light of the displacement method. Specific consideration is given to a number of different types of boundary conditions of the stretching and bending problems and the matrix modifications necessary for the introduction of these conditions to the system of equations. The additional problem-oriented language commands added to the Finite Element Analyzer for use in solving dual plate stretching and bending problems are presented in a partial user's manual intended as a supplement to the Finite Element Analyzer User's Manual.

The actual programming of the system has been done in Command Definition Language (CDL) and ICETRAN (ICES-FORTRAN) as part of ICES. Detailed system documentation (data structure, data COMMON storage, program descriptions, and program listings) is presented in the appendices.

The initial implementation of the system is in a limited form, but its flexibility and modularity allow for easy modification and sophistication at a later date.

TABLE OF CONTENTS

CHAPTER 1

Introduction and Objectives

The finite element method for the solution of structural mechanics problems has been the subject of considerable research effort during the past decade. First developed in the aircraft industry for the analysis of complex airplane fuselages, the method has spread more recently into mechanical and civil engineering work, including very recently fluid mechanics and soil mechanics. A very good brief history of the method is given by Ferrante (Ref. 1).

In the finite element method, the structural continuum is replaced by a finite number of regularly shaped three-dimensional volume elements or two-dimensional surface elements, and these elements are joined together at a finite number of common points called nodes. Then certain characteristic structural behavior quantities, usually displacement components, of any point within an element are assumed to be some polynomial function of their values at the nodes. This idealization reduces the differential equations of behavior to an approximate set of algebraic equations that are more readily solvable. Because the number of resulting unknowns and equations is very large for any meaningfully accurate application of the finite element method, the use of a digital computer for solution of the equations is a necessity. Thus the development of the finite element method has paralleled the development of the digital computer and its introduction to the solving of engineering problems. A more thorough discussion of the general finite element and its philosophy is presented by Clough (Ref. 2), Connor (Ref. 3), or Lundberg (Ref. 4).

In the analysis of plate and shell structures by the finite element method, the unknown nodal structural behavior quantities are generally taken as the displacement components associated with the two or three coordinate directions. This procedure results in the use of a stiffness matrix when the governing equations are expressed in matrix form. With the use of triangular elements and displacements that are linear functions of the nodal displacements and element coordinates, successful results are reported by Clough (Ref.2) for the problem of plate stretching, but some difficulties exist in applying the same method to the problem of plate bending (Ref. 4 and 6).

Elias (Ref. 5) has studied the mathematical duality that exists between the problems of stretching and bending of plates and has developed (Ref. 7) a finite element stress function method for plate bending that is the dual of the well-tested displacement method for plate stretching. In this dual method, nodal values of two stress functions are used as the unknown quantities, resulting in a flexibility matrix dual of the stiffness matrix of the stretching problem.

The objectives of this thesis are to review and continue the study of the dual finite element stress function method, with particular reference to formulation of all equations in matrix form and consideration of a number of different types of boundary conditions, and to implement the method as a computer system for the solution of plate bending and stretching problems.

The system takes the form of capabilities added to a general Finite Element Analyzer developed in the Department of Civil Engineering of MIT by Ferrante (Ref. 1). It is written as a problem oriented-language, which puts input and output in the language of the structural engineer and minimizes the amount of computer knowledge required of the user. The system's organization as a set of load modules will allow relatively easy modification and sophistication at a later date.

CHAPTER 2

Formulation of the System of Equations

2.1 Introduction

The finite element discretization method studied in this thesis consists of subdividing the plate structure into flat triangular elements with three nodes coinciding with the three vertices of the triangle. The two-dimensional elements represent the middle surface of the three-dimensional plate and coincide with the x-y plane of a right-handed x-y-z cartesian coordinate system.

In this chapter the geometric and material properties of the triangular element are presented first. Then the derivation of the equations for the plate stretching problem (Ref. 7) is reviewed and expressed in matrix form. Finally, the duality between the problems of plate bending and plate stretching is reviewed and the equations for the plate bending problem are obtained from those of the plate stretching problem via the duality correspondence. In both problem cases, a discussion of the solvability of the resulting sets of equations is presented.

2.2 Properties of the Orthotropic Triangular Finite Element

The triangular finite element n (figure 1) has nodes $n_1 = d$, $n_2 = e$, and $n_3 = f$ ordered counter clockwise around the element, with cartesian coordinates (x_i, y_i) for node i (i = d, e, f) referred to some global reference frame. The following notation will be adopted:

$$\begin{aligned} a_d &= -x_e + x_f \\ b_d &= -y_e + y_f \end{aligned} \tag{1}$$

Quantities a_e, a_f, b_e, and b_f are then obtained by permuting d, e, and f in Eqs 1. Note that a_i, b_i are the cartesian components of side (i), opposite node i, considered as a vector oriented counter clockwise (as shown in Fig. 1).

The area of element n is then

$$A_n = 1/2 \left| -x_d b_d - x_e b_e - x_f b_f \right| \tag{2}$$

The thickness of the element n will be taken as the average thickness of the plate in the region covered by element n:

$$h_n = \frac{1}{A_n} \iint_{A_n} h(x,y)\, dx\, dy \tag{3}$$

The material of the triangular finite element is assumed to be homogeneous, linearly elastic, and orthotropic, with axes of orthotropy coinciding with the global reference axes. Thus the following properties characterize the element:

E_x , E_y	:	Young's Moduli
ν_x , ν_y	:	Poisson's coefficients
α_x , α_y	:	coefficients of thermal expansion
G	:	shear modulus

In addition,

$$\nu_x E_x = \nu_y E_y \tag{4}$$

exists for an orthotropic material.

2.3 Derivation of the equations for the Plate Stretching Problem

The plate element n (Fig. 2) is subject to a distributed surface load vector

$$\bar{p} = p_x \bar{i} + p_y \bar{j}, \tag{5}$$

edge load intensity vectors (force/unit length)

$$\bar{N}^i = N_x^i \bar{i} + N_y^i \bar{j} \qquad (i = d, e, f), \tag{6}$$

concentrated nodal forces

$$\bar{F}_i = F_{xi} \bar{i} + F_{yi} \bar{j} \tag{7}$$

and a temperature change $\Delta T = \Delta T(x,y)$ causing initial strains which, in turn, are resisted by initial stresses

$$\begin{aligned} N_x^o &= \frac{-E_x h}{1-\nu_x\nu_y}(\varepsilon_x^o + \nu_x \varepsilon_y^o) \equiv \frac{-E_x h\Delta T}{1-\nu_x\nu_y}(\alpha_x + \nu_x\alpha_y) \\ N_y^o &= \frac{-E_y h}{1-\nu_x\nu_y}(\varepsilon_y^o + \nu_y \varepsilon_x^o) \equiv \frac{-E_y h\Delta T}{1-\nu_x\nu_y}(\alpha_y + \nu_y\alpha_x) \end{aligned} \tag{8}$$

The plate element is in equilibrium under these loads and stresses. The displacement vector

$$\bar{u}(x,y) = u(x,y)\bar{i} + v(x,y)\bar{j} \tag{9}$$

describes the displacement of a point on the middle surface of the element. If it is assumed that the displacement at any point in the element is a linear function of the displacements at the three corner nodes, i.e., that the element is in a constant state of strain, then the behavior of the plate element can be completely characterized by the six nodal variables u_i, v_i ($i = d, e, f$). Thus, the potential energy of the element, expressed as a functional of the vector displacement function $\bar{u}(x,y)$, can now be expressed as a function of six scalar variables.

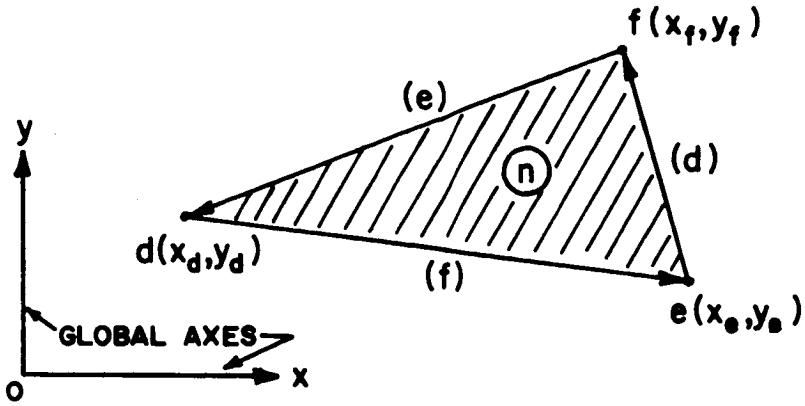


FIGURE 1. TRIANGULAR ELEMENT GEOMETRY.

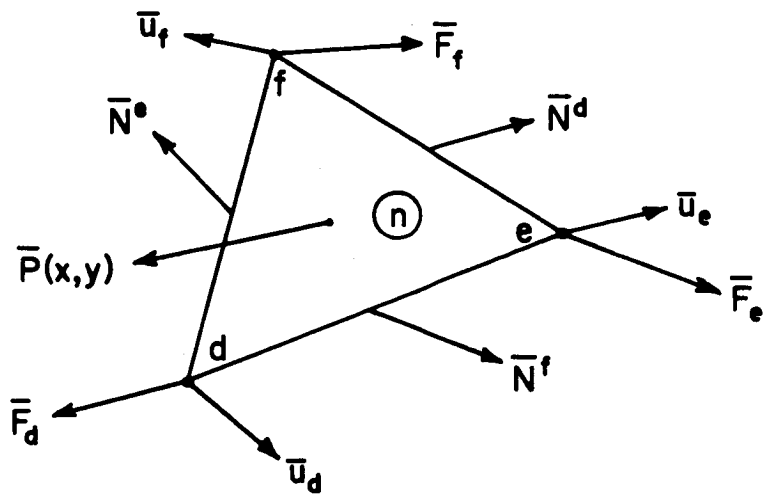


FIGURE 2. LOADS AND DISPLACEMENTS OF THE TRIANGULAR FINITE ELEMENT.

$$\pi_n = \pi_n(u_d, v_d, u_e, v_e, u_f, v_f) \tag{10}$$

$$= \text{potential energy of element n}$$

The variational principle of stationary potential energy then requires that the potential energy of element n be stationary with respect to variations of the six variables. Mathematically, this requirement takes the form

$$\left.\begin{aligned} \frac{\partial \pi_n}{\partial u_i} &= 0 \\ \frac{\partial \pi_n}{\partial v_i} &= 0 \end{aligned}\right\} \quad (i = d, e, f) \tag{11}$$

The details of this variational formulation of the plate stretching problem, including expressions for strain energy density and potential energy, are given in Ref. 7. Only the resulting expressions for Eqs. (11) will be presented here.

When the expression for π_n given in Ref. 7, Eq (49), in terms of the unknown nodal displacements, applied loadings, and geometric and material properties of element n is substituted into Eqs (11), there results a set of six force-displacement relations for the element

$$K^{xx}_{id}u_d + K^{xx}_{ie}u_e + K^{xx}_{if}u_f + K^{xy}_{id}v_d + K^{xy}_{ie}v_e + K^{xy}_{if}v_f = P^n_{xi} + R^n_{xi} + \Theta^n_{xi} + F_{xi} \quad (i = d, e, f) \tag{12}$$

$$K^{yx}_{id}u_d + K^{yx}_{ie}u_e + K^{yx}_{if}u_f + K^{yy}_{id}v_d + K^{yy}_{ie}v_e + K^{yy}_{if}v_f = P^n_{yi} + R^n_{yi} + \Theta^n_{yi} + F_{yi} \quad (i = d, e, f) \tag{13}$$

where

$$K^{xx}_{ij} = D_n\left[E_x b_i b_j + G(1-\nu_x\nu_y)\, a_i a_j\right] = K^{xx}_{ji} \tag{14}$$

$$K^{xy}_{ij} = -D_n\left[E_x \nu_x b_i a_j + G(1-\nu_x\nu_y) a_i a_j\right] = K^{yx}_{ji} \tag{15}$$

$$K^{yx}_{ij} = -D_n \left[E_y \nu_y a_i b_j + G\,(1-\nu_x \nu_y)\, b_i a_j \right] = K^{xy}_{ji} \qquad (16)$$

$$K^{yy}_{ij} = D_n \left[E_y a_i a_j + G\,(1-\nu_x \nu_y)\, b_i b_j \right] = K^{yy}_{ji} \qquad (17)$$

$$(i, j = d, e, f)$$

$$D_n = \frac{h_n}{4 A_n (1 - \nu_x \nu_y)} \qquad (18)$$

$$\left.\begin{aligned} P^n_{ri} &= \iint p_r \xi^*_i \, dA_n \qquad (19) \\ R^n_{ri} &= \sum_k \frac{1}{\ell_k} \int_0^{\ell_k} N^k_r \, s_k \, ds_k \qquad (20) \end{aligned}\right\} \begin{array}{l} i = d, e, f \\ r = x, y \end{array}$$

(k = two edges adjacent to node i; s_k oriented toward node i)

$$\left.\begin{aligned} \theta^n_{xi} &= \frac{b_i}{2A_n} \iint N^o_x \, dA_n \qquad (21) \\ \theta^n_{yi} &= \frac{a_i}{2A_n} \iint N^o_y \, dA_n \qquad (22) \end{aligned}\right\} i = d, e, f$$

Thus, P^n_{xi}, P^n_{yi} are the generalized nodal forces due to the distributed surface load; R^n_{xi}, R^n_{yi} are the generalized nodal forces due to the edge load intensities; and θ^n_{xi}, θ^n_{yi} are the generalized nodal forces due to the temperature change. The total generalized nodal loads (right hand sides of Eqs. (12) and (13))will be denoted by S^n_{xi} and S^n_{yi} in what follows.

Let $\underset{\sim}{S}^n_i = \left\{ S^n_{xi} \quad S^n_{yi} \right\}$ be the total generalized nodal load column matrix. Let $\underset{\sim}{U}_i = \left\{ u_i \quad v_i \right\}$ be the nodal displacement matrix (i = d, e, f). Then Eqs (12) and (13) can be expressed as three matrix equations

$$\underset{\sim}{K}^n_{id} U_d + \underset{\sim}{K}^n_{ie} U_e + \underset{\sim}{K}^n_{if} U_f = \underset{\sim}{S}^n_i \quad (i = d, e, f) \qquad (23)$$

where

$$\underset{\sim}{K}^n_{ij} = \left[\begin{array}{c|c} K^{xx}_{ij} & K^{xy}_{ij} \\ \hline K^{yx}_{ij} & K^{yy}_{ij} \end{array} \right] \qquad (i,j = d, e, f) \qquad (24)$$

*Symbols ξ_i are the dimensionless triangular coordinates. They are explained on page 5 of Ref. 7.

$\underset{\sim}{K}^n_{ij}$ is not symmetric, because $K^{xy}_{ij} \neq K^{yx}_{ij}$, but $\underset{\sim}{K}^n_{ij} = (\underset{\sim}{K}^n_{ji})^T$. The superscript n denotes the element to which $\underset{\sim}{K}^n_{ij}$ belongs.

Now consider a typical node i in a finite element representation of a continuous plate (fig. 3). Node i is common to elements k, l, m, n and p, and is connected through these elements to nodes q, r, s, t and w. Equation (23) can be considered as the contribution of element n to the total generalized nodal force at node i, if i, s, and t replace d, e, and f. Thus the total generalized nodal force at node i due to all elements incident upon it would be

$$\underset{\sim}{S}_i = \sum_j (\underset{\sim}{S}^j_i - \underset{\sim}{F}_i) + \underset{\sim}{F}_i \quad (j = k, l, m, n, p) \tag{25}$$

The definition of $\underset{\sim}{S}^j_i$ includes the concentrated nodal force $\underset{\sim}{F}_i$. Since this force acts only once at each node, it must be subtracted from $\underset{\sim}{S}^j_i$ before the summation in Eq (25) and added back in after the summation.

The total matrix force-displacement relation for node i becomes

$$\begin{aligned} &(\sum_j \underset{\sim}{K}^j_{ii})\,\underset{\sim}{U}_i + (\underset{\sim}{K}^1_{ir} + \underset{\sim}{K}^m_{ir})\,\underset{\sim}{U}_r + (\underset{\sim}{K}^m_{is} + \underset{\sim}{K}^n_{is})\,\underset{\sim}{U}_s \\ &(\underset{\sim}{K}^n_{it} + \underset{\sim}{K}^p_{it})\,\underset{\sim}{U}_t + (\underset{\sim}{K}^p_{iw} + \underset{\sim}{K}^k_{iw})\,\underset{\sim}{U}_w + (\underset{\sim}{K}^k_{iq} + \underset{\sim}{K}^1_{iq})\,\underset{\sim}{U}_q = \underset{\sim}{S}_i \end{aligned} \tag{26}$$

Eqs (26) can be generalized to apply to any node in the plate. The system of equations resulting from the application of Eqs (26) to every node of the structure is (in supermatrix form)

$$\underset{\sim}{K}\ \underset{\sim}{U} = \underset{\sim}{S} \tag{27}$$

or explicitly

$$\begin{bmatrix} \underset{\sim}{K}_{11} & \underset{\sim}{K}_{12} & \cdots & \underset{\sim}{K}_{1n} \\ \underset{\sim}{K}_{21} & \underset{\sim}{K}_{22} & \cdots & \underset{\sim}{K}_{2n} \\ \vdots & \vdots & & \vdots \\ \cdots & \underset{\sim}{K}_{ij} & \cdots & \cdots \\ \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} \underset{\sim}{U}_1 \\ \underset{\sim}{U}_2 \\ \vdots \\ \underset{\sim}{U}_j \\ \vdots \end{bmatrix} = \begin{bmatrix} \underset{\sim}{S}_1 \\ \underset{\sim}{S}_2 \\ \vdots \\ \underset{\sim}{S}_j \\ \vdots \end{bmatrix} \quad 2n$$

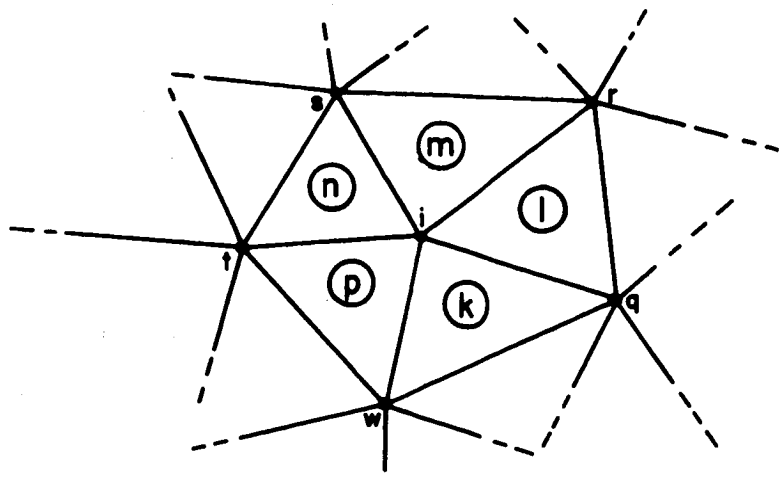


FIGURE 3. A TYPICAL NODE (i) IN A TRIANGULAR FINITE ELEMENT REPRESENTATION OF A PLATE.

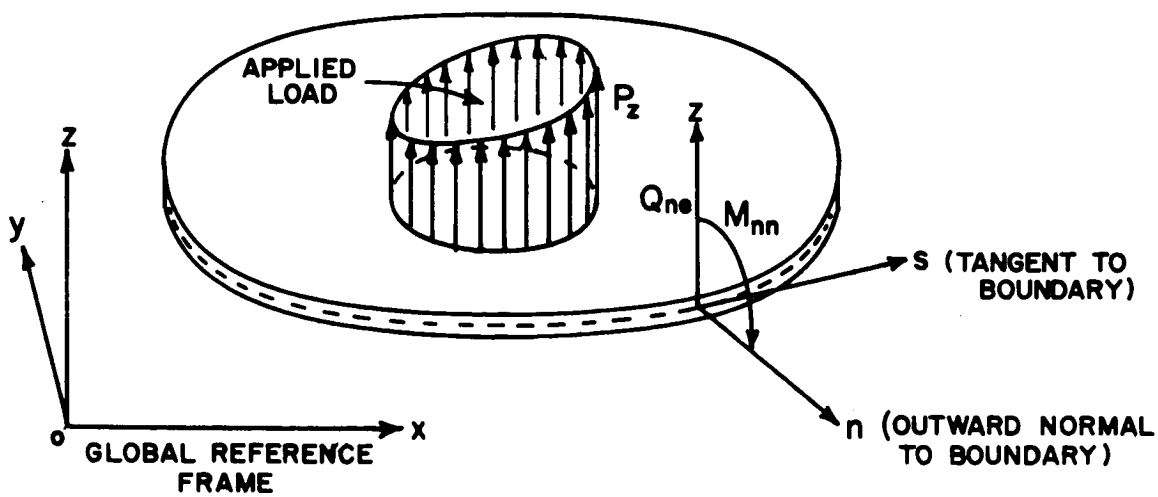
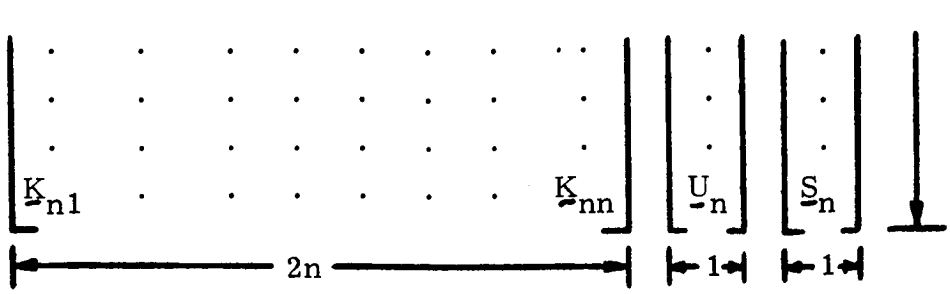


FIGURE 4. PLATE BENDING PROBLEM NOTATION.



where n now equals the total number of nodes in the finite element representation. The assembly of the supermatrices $\underset{\sim}{K}$ and $\underset{\sim}{S}$ follows the well-established procedure of the direct stiffness method (which follows from Eqs (25):

1) For element n, with nodes n_1, n_2, and n_3:

   a) $\underset{\sim}{K}^n_{n_i n_j}$ in super-row n_i, super-column n_j of $\underset{\sim}{K}$ $(i,j = 1,2,3)$ (28)

   b) $(\underset{\sim}{S}^n_{n_i} - \underset{\sim}{F}_{n_i})$ in super-row n_i of $\underset{\sim}{S}$ $(i = 1,2,3)$ (29)

   Step 1) is repeated for every element in the structure.

2) $\underset{\sim}{F}_i$ in super-row i of $\underset{\sim}{F}$ $(i = 1,2,3.....n)$ (30)

   Since $\underset{\sim}{K}^n_{ij} = (\underset{\sim}{K}^n_{ji})^T$, the resulting stiffness matrix $\underset{\sim}{K}$ is symmetric, and thus only half of it (the diagonal plus elements below the diagonal) need be assembled.

The system of equations (27) is singular, i.e., there exists a non-trivial solution to the system where $\underset{\sim}{S} = \underset{\sim}{0}$, and thus $|\underset{\sim}{K}| = 0$. The non-trivial solutions $\underset{\sim}{U}^o$ are the nodal displacements of a rigid-body motion, which was not suppressed in the preceeding development. To fix the plate against rigid-body motion, the displacements of some node r (u_r, v_r) plus the rotation about that node must be known. Knowing the rotation is equivalent to knowing the displacement of another node s in the direction perpendicular to the line joining r and s. Thus three displacement components must be known to suppress rigid-body motion of the plate. These displacements are then the minimum boundary conditions that must be specified in order to solve Eqs (27). Once they are specified, they may be moved over to the right hand side (known side), and the number of unknowns reduced by three. Thus three equations from among Eqs (27) may be eliminated. The resulting system of 2n-3 equations is non-singular and may be solved by inverting the

reduced $\underset{\sim}{K}$ matrix or any other convenient method.

The generalized nodal forces $\underset{\sim}{R}^n_{xi}$ and $\underset{\sim}{R}^n_{yi}$ resulting from edge stress resultant intensities N_{nx} and N_{ny} contribute nothing to the total generalized nodal force $\underset{\sim}{S}_i$ from an interior edge (an edge common to two elements) because the stress resultants are equal in magnitude but opposite in direction for the two elements joined along the edge. In the initial assembly of the $\underset{\sim}{S}$ matrix, the contributions of $\underset{\sim}{R}^n_i$ can be ignored at <u>all</u> nodes because the contribution is zero for an interior edge and the contribution for a boundary edge will be added later during the explicit treatment of boundary conditions. Thus, <u>initially,</u> $\underset{\sim}{S}_i = \underset{\sim}{P}_i + \underset{\sim}{\theta}_i + \underset{\sim}{F}_i$, $(i = 1, 2, \ldots . n)$.

2.4 Duality Between the Problem of Plate Stretching and Plate Bending

Consider a plate in the x-y plane (Fig. 4), in equilibrium under an applied surface load vector intensity

$$\bar{p} = p_z(x, y)\,\bar{k} \tag{31}$$

boundary effective shears of vector intensity

$$\bar{Q} = Q_{ne}(s)\,\bar{k} \tag{32}$$

and boundary bending stress couples of vector intensity

$$\bar{M} = M_{nn}(s)\,\bar{s}\,. \tag{33}$$

For small displacements, this system of loads will produce primarily bending in the plate, with negligible stretching.

The solution of the plate bending problem can be considered as composed of two parts: 1) a particular solution of the equilibrium equations that equilibrates the applied surface load without necessarily satisfying compatability or the boundary conditions, and 2) a self-equilibrating homogeneous solution that compensates for the particular solution in such a way that the total or general solution (sum of homogeneous and particular) satisfies equilibrium, compatibility, and the boundary conditions.

The equilibrium equations for plate bending are

$$M_{xx,x} + M_{yx,y} - Q_x = 0 \tag{34}$$

$$M_{xy,x} + M_{yy,y} - Q_y = 0 \tag{35}$$

$$Q_{x,x} + Q_{y,y} + p_z = 0 \tag{36}$$

where $_{,r}$ denotes differentiation with respect to r, the M's are stress couples, and the Q's are shears, as defined in Fig. 5.

A particular solution of Eqs. 34-36, denoted by superscript p, is taken as

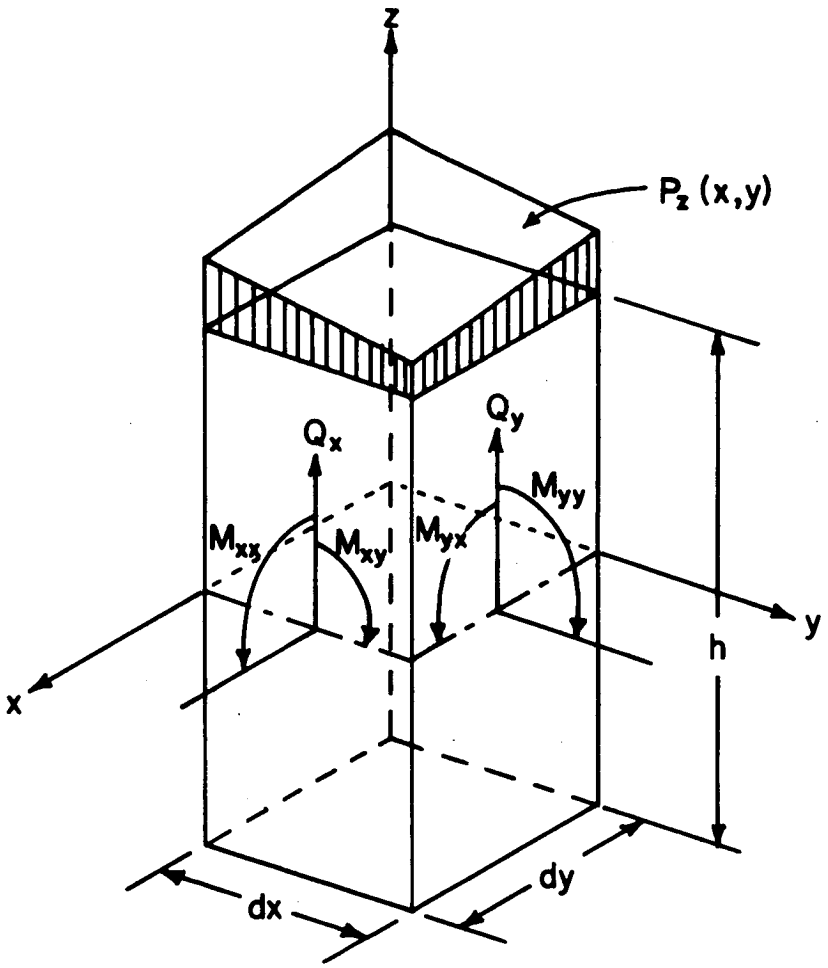


FIGURE 5. DEFINITION OF SHEARS AND STRESS COUPLES ACTING ON THE DIFFERENTIAL PLATE ELEMENT.

$$M^p_{xy} = M^p_{yx} = 0 \tag{37}$$

$$M^p_{xx} = -D_x(K_y + \nu_x K_x) \tag{38}$$

$$M^p_{yy} = -D_y(K_x + \nu_y K_y) \tag{39}$$

$$Q^p_x = M^p_{xx,x} \tag{40}$$

$$Q^p_y = M^p_{yy,y} \tag{41}$$

where

$$D_x = \frac{E_x h^3}{12(1 - \nu_x \nu_y)} \tag{42}$$

$$D_y = \frac{E_y h^3}{12(1 - \nu_x \nu_y)} \tag{43}$$

and K_x, K_y are particular solution functions satisfying the differential equation.

$$\left[D_x(K_y + \nu_x K_x)\right]_{,xx} + \left[D_y(K_y + \nu_y K_y)\right]_{,yy} - p_z = 0 \tag{44}$$

The stress couples and shears of Eqs. 37-41 are those that would occur in two families of strips parallel to the coordinate axes, arbitrarily supported and acting independently of each other. The load p_z may then be divided arbitrarily between the two families of strips. The reason for introducing the two functions K_x and K_y will become apparent in what follows.

The variables of the bending problem can be expressed as sums of the particular solution portion and the homogeneous solution portion; for example

$$M_{xx} = M^p_{xx} + M^*_{xx} \tag{45}$$

where * indicates a quantity associated with the homogeneous solution.

Representing the variables in this way, a duality exists between the equations of plate stretching and plate bending (Ref. 7). In particular, the finite element displacement method of solution of the plate stretching problem, developed in the previous section, has a dual stress function method of solution for the bending problem. The force-displacement relations of the stretching problem become rotation-stress function relations of the bending problem upon substitution of the corresponding dual variables and constants. The correspondence of this duality is summarized in Table 1.

The resulting system of equations for solution of the plate bending problem becomes, in supermatrix form

$$\underset{\sim}{K}' \underset{\sim}{U}' = \underset{\sim}{S}' \tag{46}$$

where

$\underset{\sim}{K}'$ = bending "flexibility" matrix dual of the stretching stiffness matrix (47)

$\underset{\sim}{U}'$ = column matrix of nodal values of the stress functions U and V (48)

$\underset{\sim}{S}'$ = column matrix of total generalized nodal rotations, dual of the generalized nodal force matrix (49)

This system is also singular and the non-trivial solution $\underset{\sim}{U}'^{o}$ of the homogeneous system ($\underset{\sim}{S}' = 0$) are dual of the rigid-body displacements of the stretching problem. Just as the rigid-body displacement yields zero strains, the solutions $\underset{\sim}{U}'^{o}$ for the stress functions yield zero stress couples and shears.

TABLE I

Duality Between the Problems of Stretching and Bending

Stretching	Bending
Equations:	
Equilibrium	Compatibility
Compatibility	Equilibrium
Stress-Strain	Stress-Strain
Strain-Displacement	Stress-Stress Function
Stress-Stress Function	Curvature-Displacement
Material Properties:	
$E_x h$, $E_y h$, Gh	$-D_y^{-1}$, $-D_x^{-1}$, $-(Gh^3/3)^{-1}$
ν_x, ν_y	$-\nu_x$, $-\nu_y$
Problem Variables:	
u, v	U, V (stress functions)
N_x^o, N_y^o	$-\varkappa_y^o$, $-\varkappa_x^o$ (thermal curvatures)
p_x, p_y	$K_{x,x}$, $K_{y,y}$
N_{xx}, N_{xy}, N_{yy}	$-\varkappa_{yy}^*$, $\varkappa_{xy}$, $-\varkappa_{xx}^*$ (cartesian curvatures)
ϵ_{xx}, ϵ_{xy}, ϵ_{yy}	M_{yy}^*, $-M_{xy}$, M_{xx}^*
N_{nx}, N_{ny}	$-\varkappa_{sy}^*$, $\varkappa_{sx}^*$ (boundary curvatures)
ϵ_{ss}, $\varkappa$ (boundary strain and in-plane curvature)	M_{nn}^*, Q_{ne}^*
Matrices:	
$\underset{\sim}{K}_{ij}$, $\underset{\sim}{P}_i$, $\underset{\sim}{R}_i$, $\underset{\sim}{\Theta}_i$, $\underset{\sim}{U}_i$ ($i = 1,2,\ldots,n$)	$\underset{\sim}{K}'_{ij}$, $\underset{\sim}{P}'_i$, $\underset{\sim}{R}'_i$, $\underset{\sim}{\Theta}'_i$, $\underset{\sim}{U}'_i$ ($i = 1,2,\ldots,n$)

CHAPTER 3

3.1 Introduction of Boundary Conditions for the Plate Stretching Problem

1) Stress Boundary Conditions:

Known edge load intensities N_{nx} and N_{ny} contribute to the generalized nodal force at the boundary nodes. Thus at a node i joining boundary segments (boundary element edges) m and n

$$R_{ri} = \frac{1}{\ell_m} \int_o^m N_r^m \, s_m \, ds_m + \frac{1}{\ell_n} \int_o^n N_r^n \, s_n \, ds_n \qquad (50)$$

($r = x, y$; s_m, s_n oriented towards node i)

Then $\underset{\sim}{R}_i = \{R_{xi} \; R_{yi}\}$ is added to $\underset{\sim}{S}_i$ for every node i along portions of the boundary on which edge load intensities are known.

If the acutal edge load intensities, functions of positions along the boundary, are approximated by a linear variation between boundary nodes, then they may be completely specified by their values at the boundary nodes, N_{xi} and N_{yi}. Using this approximation, the integrals of Eqs.(50) are evaluated to give

$$R_{ri} = \frac{\ell m}{6} (N_{rj} + 2N_{ri}) + \frac{\ell n}{6} (N_{rh} + 2N_{ri}) \qquad (51)$$

where j and h are the nodes at the other ends of segments m and n, respectively.

2) Displacement Boundary Conditions:

If the two componenets of displacements at a node i (u_i, v_i) are known, then there are two less unknown nodal displacements and the corresponding two equations at node i (Eqs(26)) can be deleted from the system of Eqs (27). Also, terms involving $\underset{\sim}{U}_i$ in the other equations of system (27) are known and can be moved to the right hand side and added to $\underset{\sim}{S}_i$. The modifications to the stiffness and load matrices for the introduction of known pairs of displacements at a node i are then accomplished as follows:

a) in $\underset{\sim}{K}$:

$$\underset{\sim}{K}_{ij} \Rightarrow \underline{0} \qquad j \neq i, \quad j = 1, 2, \ldots n \tag{52}$$

$$\underset{\sim}{K}_{ji} \Rightarrow \underline{0} \tag{53}$$

$$\underset{\sim}{K}_{ii} \Rightarrow \underline{I}_2 \quad (2 \times 2 \text{ unit matrix}) \tag{54}$$

b) in $\underset{\sim}{S}$:

$$\underset{\sim}{S}_j \Rightarrow \underset{\sim}{S}_j - \underset{\sim}{K}_{ji}\,\underset{\sim}{U}_i \qquad j \neq i, \quad j = 1, 2, \ldots n \tag{55}$$

$$\underset{\sim}{S}_i \Rightarrow \underset{\sim}{U}_i \tag{56}$$

where $\Rightarrow$ means "is replaced by."

The matrix multiplications shown above and in later sections of this chapter are used as notational devices to arrive at a more clear, concise presentation of the theory. In the actual computer solution of such problems, the matrix multiplications are usually not performed explicitly, and in fact many of the matrices defined in this chapter are not actually used explicitly in the computer.

3) Mixed Boundary Conditions:

Consider the case of a boundary node i at which one component of displacement is known in a direction r (u_r) and one component of edge stress resultant is known in the direction perpendicular to r along the edge segments adjacent to node i (Fig. 6). The angle ϕ relates the r direction to the global x-axis, positive from x to r. The treatment of these mixed boundary conditions follows the well-known procedure given below:

a) Define five matrices $\underset{\sim}{E}_i$, $\underset{\sim}{G}_i$, $\underset{\sim}{u}_i^*$, $\underset{\sim}{R}^i$, and $\underset{\sim}{R}_i^*$,

$$\underset{\sim}{E}_i = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad \underset{\sim}{G}_i = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad \underset{\sim}{u}_i^* = \begin{Bmatrix} u_{ri} \\ o \end{Bmatrix} \tag{57}$$

$$\underset{\sim}{R}^i = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \qquad \underset{\sim}{R}_i^* = \begin{Bmatrix} 0 \\ R_{qi} \end{Bmatrix} \tag{58}$$

where q is the direction perpendicular to r, and R_{qi} is determined from

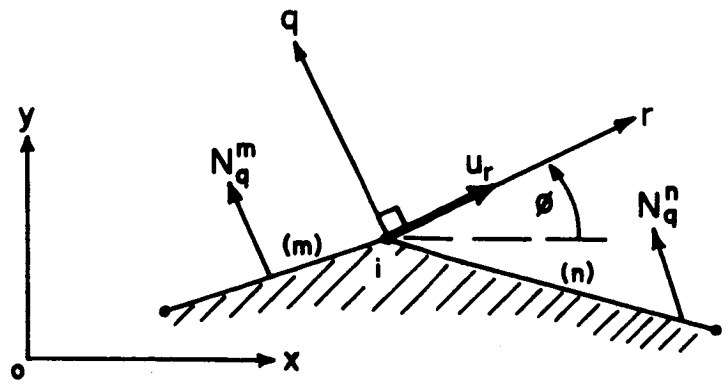


FIGURE 6. NOTATION FOR MIXED BOUNDARY CONDITIONS.

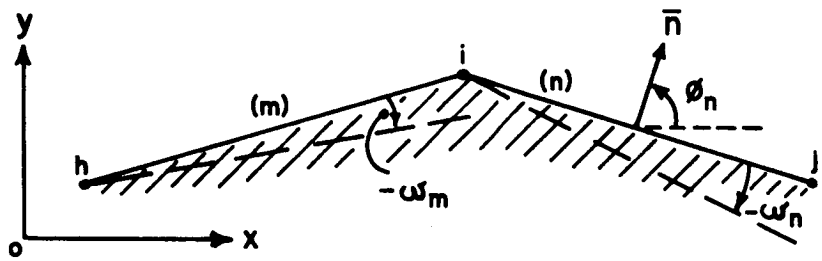


FIGURE 7. NOTATION FOR THE EDGE BEAM BOUNDARY.

Eq. (50) or Eq. (51) with r replaced by q.

b) Perform the following modifications upon the matrices $\underset{\sim}{K}$ and $\underset{\sim}{S}$:

$$\text{i)}\quad \underset{\sim}{K}_{ji} \Rightarrow \underset{\sim}{K}_{ji}\underset{\sim}{R}^{i}\underset{\sim}{E}_{i} \qquad (j = 1, 2, \ldots, n \neq i) \tag{59}$$

$$\underset{\sim}{S}_{j} \Rightarrow \underset{\sim}{S}_{j} - \underset{\sim}{K}_{ji}\underset{\sim}{R}^{i}\underset{\sim}{u}_{i}^{*} \tag{60}$$

$$\text{ii)}\quad \underset{\sim}{K}_{ii} \Rightarrow (\underset{\sim}{E}_{i}\underset{\sim}{R}^{i,\,T})\,\underset{\sim}{K}_{ii}(\underset{\sim}{R}^{i}\underset{\sim}{E}_{i}) + \underset{\sim}{G}_{i} \tag{61}$$

$$\underset{\sim}{S}_{i} \Rightarrow \underset{\sim}{R}_{i}^{*} + \underset{\sim}{E}_{i}\underset{\sim}{R}^{i,\,T}(\underset{\sim}{S}_{i} - \underset{\sim}{K}_{ii}\underset{\sim}{R}^{i}\underset{\sim}{u}_{i}^{*}) + \underset{\sim}{u}_{i}^{*} \tag{62}$$

$$\text{iii)}\quad \underset{\sim}{K}_{ij} \Rightarrow (\underset{\sim}{E}_{i}\underset{\sim}{R}^{i,\,T})\,\underset{\sim}{K}_{ij} \tag{63}$$

$$(j = 1, 2, \ldots, n \neq i)$$

The solution of Eqs (27) will then yield $\underset{\sim}{U}_{i} = \{u_{ri}\ u_{qi}\}$ for the displacements at node i. To obtain the displacements referred to the global axes, $\underset{\sim}{U}_{i}$ must be multiplied by $\underset{\sim}{R}^{i}$, the rotation matrix from the r-q axes to the x-y axes.

4) Elastic Boundary Supports:

If a boundary of the plate is supported elastically, the stress resultants on the boundary edge will be functions of the unknown edge displacements. Thus the generalized nodal forces due to these edge stresses will be functions of the unknown nodal displacements and must be moved to the left hand side of Eqs. (27).

If the elasticity of the boundary support is specified by k_{xx}, k_{xy}, and k_{yy} (all functions of position along the boundary) such that

$$N_{nx} = k_{xx}(u^{s} - u) + k_{xy}(v^{s} - v) \tag{64}$$

$$N_{ny} = k_{yx}(u^{s} - u) + k_{yy}(v^{s} - v) \tag{65}$$

where u^{s} and v^{s} are the known displacements of the elastic boundary support, then the generalized nodal forces at each node i along the elastic boundary become

$$R_{xi} = \sum_{k=h,i,j} k_{ik}^{xx}(u_k^s - u_k) + \sum_{k=h,i,j} k_{ik}^{xy}(v_k^s - v_k) \tag{66}$$

$$R_{yi} = \sum k_{ik}^{yx}(u_k^s - u_k) + \sum k_{ik}^{yy}(v_k^s - v_k) \tag{67}$$

where h and j are the nodes before and after i on the boundary. In matrix form, Eqs. (66) and (67) become

$$\underline{R}_i = \underline{k}_{ih}(\underline{U}_h^s - \underline{U}_h) + \underline{k}_{ii}(\underline{U}_i^s - \underline{U}_i) + \underline{k}_{ij}(\underline{U}_j^s - \underline{U}_j) \tag{68}$$

where

$$\underline{k}_{ir} = \left[\begin{array}{c|c} k_{ir}^{xx} & k_{ir}^{xy} \\ \hline k_{ir}^{yx} & k_{ir}^{yy} \end{array}\right] \quad (r = h, i, j) \tag{69}$$

$$k_{ir}^{xx} = \frac{1}{\ell_m^2}\int_o^{\ell_m} k_{xx} S_m^2 \, ds_m + \frac{1}{\ell_n^2}\int_o^{\ell_n} k_{xx} S_n^2 ds_n \tag{70}$$

$$k_{ih}^{xx} = \frac{1}{\ell_m^2}\int_o^{\ell_m} k_{xx} S_m (\ell_m - S_m)\, dS_m \tag{71}$$

$$k_{ij}^{xx} = \frac{1}{\ell_n^2}\int_o^{\ell_n} k_{xx} S_n (l_n - S_n)\, dS_n \tag{72}$$

and m and n are the segments before and after node i; (s_m and s_n are oriented towards node i). Quantities k_{ij}^{xy}, k_{ij}^{yx}, and k_{ij}^{yy} are obtained from Eqs. (70) - (72) by using in turn k_{xy}, k_{yx}, and k_{yy} in the integrals.

Eqs. (68) require the following matrix modifications to $\underline{K}$ and $\underline{S}$ for each row i along the elastic boundary:

a) in $\underline{K}$:

$$\underline{K}_{ih} \Rightarrow \underline{K}_{ih} + \underline{k}_{ih} \tag{73}$$

$$\underline{K}_{ii} \Rightarrow \underline{K}_{ii} + \underline{k}_{ij} \tag{74}$$

$$\underset{\sim}{K}_{ij} \Rightarrow \underset{\sim}{K}_{ij} + \underset{\sim}{k}_{ij} \tag{75}$$

b) in $\underset{\sim}{S}$:

$$\underset{\sim}{S}_i \Rightarrow \underset{\sim}{S}_i + \underset{\sim}{k}_{ih}\underset{\sim}{U}^s_h + \underset{\sim}{k}_{ii}\underset{\sim}{U}^s_i + \underset{\sim}{k}_{ij}\underset{\sim}{U}^s_j \tag{76}$$

5) Plate Bounded by an Edge Beam:

If the plate is supported by an edge beam, account must be taken of the beam potential energy in the variational formulation of Eqs (27).

If the beam is loaded externally by stress resultants $\bar{N}_x$ and $\bar{N}_y$ per length of edge beam, they give rise to generalized nodal forces $\underset{\sim}{\bar{F}}_i = \left\{ \bar{F}_{xi} \quad \bar{F}_{yi} \right\}$ at each node i along the edge beam, as determined by equations like Eqs (50). A portion of these nodal forces is resisted by the edge beam and the remainder is transmitted to the plate elements. If the portion resisted by the edge beam is denoted by $\underset{\sim}{\Delta F}_i = \left\{ \Delta F_{xi} \quad \Delta F_{yi} \right\}$ at a node i, then $\underset{\sim}{R}_i$, the portion transmitted to the plate, is given by

$$\underset{\sim}{R}_i = \underset{\sim}{F}_i - \underset{\sim}{\Delta F}_i \tag{77}$$

Expressions for $\underset{\sim}{\Delta F}_i$ in terms of the nodal displacements and the beam material properties are determined from variations of the beam potenetial energy with respect to the nodal displacements. Account must be taken of the strain energy due to bending, which is expressed in terms of the curvature of the edge beam. In the piece-wise linear boundary idealization used here, curvature does not actually exist and must be interpreted instead as the difference between side rotations of two adjacent boundary segments. The rotation of a segment n connceting nodes i and j is given by

$$W_n = \frac{1}{\ell_n} \left[- (u_j - u_i) \cos\phi_n - (v_j - v_i) \sin\phi_n \right] \tag{78}$$

where ϕ_n is the angle from the x-axis to the outward boundary normal vector n at segment n (Fig. 7). The curvature at node i is then expressed as the difference of rotations of adjacent segments m and n

$$\chi_i = (W_m - W_n) \left(\frac{2}{\ell_m + \ell_n} \right) \tag{79}$$

Introducing the notation

$$d_k = \frac{(EI)_{k-1} + (EI)_k}{(\ell_k + \ell_{k-1})} \tag{80}$$

$$\alpha_k = E_k A_k \ell_k \tag{81}$$

$$s_k = \frac{\sin\phi_k}{\ell_k} = \frac{-a_k}{\ell_k^2} \tag{82}$$

$$c_k = \frac{\cos\phi_k}{\ell_k} = \frac{b_k}{\ell_k^2} \tag{83}$$

where A_k is the cross-section area, E_k the Young's modulus, I_k the moment of inertia about the z-axis, and ℓ_k the length of edge beam segment k, which connects node k to node k +1, the following equivalent stiffness factors (Ref. 7) can be used to represent the beam behavior at a node i (Fig. 8):

$$\left.\begin{aligned}
k_{ii}^{xx} &= \alpha_h S_h^2 + \alpha_i S_i^2 + d_i (C_h + C_i)^2 + d_h C_h^2 + d_j d_i^2 \\
k_{ih}^{xx} &= \alpha_h S_h^2 - d_i C_h (C_h + C_i) - d_h C_h (C_h + C_g) \\
k_{ig}^{xx} &= a_i S_i^2 - d_i C_i (C_h + C_i) - d_j d_i (C_i + C_j) \\
k_{ig}^{xx} &= d_h C_g C_h \\
k_{ik}^{xx} &= d_j C_i C_j \\
k_{ii}^{xy} &= -\alpha_h S_h C_h - \alpha_i S_i C_i + d_i (C_i + C_h)(S_i + S_h) \\
&\quad + d_h S_h C_h - d_j S_i C_i \\
k_{ih}^{xy} &= \alpha_h S_h C_h - d_i S_h (C_h + C_i) - d_h C_h (S_h + S_g)
\end{aligned}\right\} \tag{84}$$

$$\left.\begin{aligned} k_{ij}^{xy} &= \alpha_i S_i C_i - d_i S_i (C_h + C_i) - d_j C_i (S_i + S_j) \\ k_{ig}^{xy} &= d_h S_g C_h \\ k_{ik}^{xy} &= d_j S_j C_i \end{aligned}\right\} \qquad (84)$$

Quantities k_{iq}^{yx} and k_{iq}^{yy} ($q = g,h,i,j,k$) are obtained from Eqs (84) by interchanging x and y and s_q and c_q. Using these stiffness factors, the generalized nodal forces become

$$\underset{\sim}{\Delta F}_i = \underset{\sim}{F}_i^o + \sum_{n=g,h,i,j,k} \underset{\sim}{k}_{in} \underset{\sim}{U}_n \qquad (85)$$

where

$$\underset{\sim}{k}_{in} = \left[\begin{array}{c|c} k_{in}^{xx} & k_{in}^{xy} \\ \hline k_{in}^{yx} & k_{in}^{yy} \end{array}\right] \qquad (86)$$

and $\underset{\sim}{F}_i^o = \left\{F_{xi}^o \quad F_{yi}^o\right\}$ are generalized nodal forces due to initial thermal strains in the beam, given by (Ref. 7)

$$\begin{aligned} F_{xi}^o = {} & \alpha_n S_n \epsilon_h^o - \alpha_i S_i \epsilon_i^o - \tfrac{1}{2} d_i (C_h + C_i)(\ell_h + \ell_i)\chi_i^o \\ & + \tfrac{1}{2} d_h C_h (\ell_g + \ell_h)\chi_n^o + \tfrac{1}{2} d_j C_i (\ell_i + \ell_j)\chi_j^o \end{aligned} \qquad (87)$$

$$\begin{aligned} F_{xi}^o = {} & -\alpha_h C_h \epsilon_h^o + \alpha_i C_i \epsilon_i^o - \tfrac{1}{2} d_i (S_h + S_i)(\ell_h + \ell_i)\chi_i^o \\ & + \tfrac{1}{2} d_h S_h (\ell_g + \ell_n)\chi_h^o + \tfrac{1}{2} d_j S_i (\ell_i + \ell_j)\chi_j^o \end{aligned} \qquad (88)$$

where ϵ_i^o is an axial thermal strain due to a uniform temperature change in the beam and χ_i^o is a curvature in the x-y plane due to a temperature differential between the inner and outer faces of the beam.

Eqs. (85) result in the following modifications to $\underset{\sim}{K}$ and $\underset{\sim}{S}$ for each node i on the edge beam boundary:

a) row of i of $\underset{\sim}{K}$:

$$\underset{\sim}{K}_{iq} \Rightarrow \underset{\sim}{K}_{iq} + \underset{\sim}{k}_{iq} \quad (q = g,h,i,j,k) \tag{89}$$

b) row of i of $\underset{\sim}{S}$:

$$\underset{\sim}{S}_i \Rightarrow \underset{\sim}{S}_i - \underset{\sim}{F}_i^o + \underset{\sim}{\bar{F}}_i \tag{90}$$

6) Strain Boundary Conditions:

Consider a portion of the plate boundary along which externsional strain and in-plane curvature are known. In the finite element idealization used here, displacements vary linearly between nodes. Thus the boundary extensional strain (ϵ_{ss}) is constant in any boundary segment and each segment remains straight after deformation. As mentioned during consideration of edge beams, the curvature is then defined only at each node and is the difference between the rotations of the two adjacent boundary segments. Thus strain boundary conditions take the form of an extensional strain prescribed for each segment and a curvature prescribed for each node. Then

$$\epsilon_n = -\sin\phi_n (u_{n+1}-u_n) + \cos\phi_n (v_{n+1}-v_n) \tag{91}$$

for a segment n joining nodes n and n+1. Using Eqs (78) and (79), the curvature at node n becomes

$$\times_n = \frac{2}{\ell_n + \ell_{n-1}} \left\{ \frac{1}{\ell_n} \left[-(u_{n+1}-u_n)\cos\phi_n - (v_{n+1}-v_n)\sin\phi_n \right] - \frac{1}{\ell_{n-1}} \left[-(u_n-u_{n-1})\cos\phi_{n-1} - (v_n-v_{n-1})\sin\phi_{n-1} \right] \right\} \tag{92}$$

where segment n-1 joins node n-1 to node n.

If the total number of nodes along the strain portion of the boundary (including the two end nodes) is s, then there are 2s unknown nodal displacements. One equation like (91) can be written for each segment and one equation like (92) can be written for each node except the two end nodes, for a total of 2s-3 equations. The remaining three conditions needed to solve the strain boundary portion are usually one of the following groups:

a) Specification of the two force resultants and moment of the boundary forces (along the boundary portion being considered) with regard to some point.

b) Specification of the three components of rigid body motion of the boundary portion (2 displacements at a node plus the rotation of an adjacent edge segment).

c) Specification of two displacement components of a point p not necessarily on the boundary line plus specification of the resultant moment of the boundary forces with regard to the same point p.

This third specification group is used when the plate is attached to a rigid material body that is pinned at some point, i.e., fixed to rotate about some point not necessarily on the plate-rigid material interface (Fig. 9).

It is noted that the above equations for solving the boundary portion, which will replace the force-displacement relations for all nodes i along the strain boundary portion, are compatibility or strain-displacement relations rather than the equilibrium equations upon which Eqs (27) are based. Introduction of strain boundary conditions into the system of Eqs (27) will result in certain rows being replaced without replacing the corresponding columns, and thus the "stiffness" matrix $\underset{\sim}{K}$ will no longer be symmetrical. The matrix modifications for the introduction of strain boundary conditions are now developed.

Consider the strain portion of the boundary to have nodes numbered consecutively from 1 to s in the +s direction, with segment (i) following node i. The +s direction is the direction traversed along the boundary keeping the outward normal on the right (Fig. 10). At each node i (i=1,s) Eq (91) for segment (i) can be combined with Eq (92) for node i to give the matrix equation

$$-\underset{\sim}{J}_{i-1}\,\underset{\sim}{U}_{i-1} + (\underset{\sim}{J}_{i-1} + \underset{\sim}{H}_i)\,\underset{\sim}{U}_i - \underset{\sim}{H}_i\,\underset{\sim}{U}_{i+1} = \underset{\sim}{\epsilon}_i \tag{93}$$

where

$$\underset{\sim}{J}_n = \frac{1}{\ell_n}\left[\begin{array}{c|c} 0 & 0 \\ \hline \cos\phi_n & \sin\phi_n \end{array}\right], \qquad \underset{\sim}{H}_n = \left[\begin{array}{c|c} \dfrac{\sin\phi_n}{\cos\phi_n}\,\ell_n & \dfrac{-\cos\phi_n}{\sin\phi_n}\,\ell_n \end{array}\right] \tag{94}$$

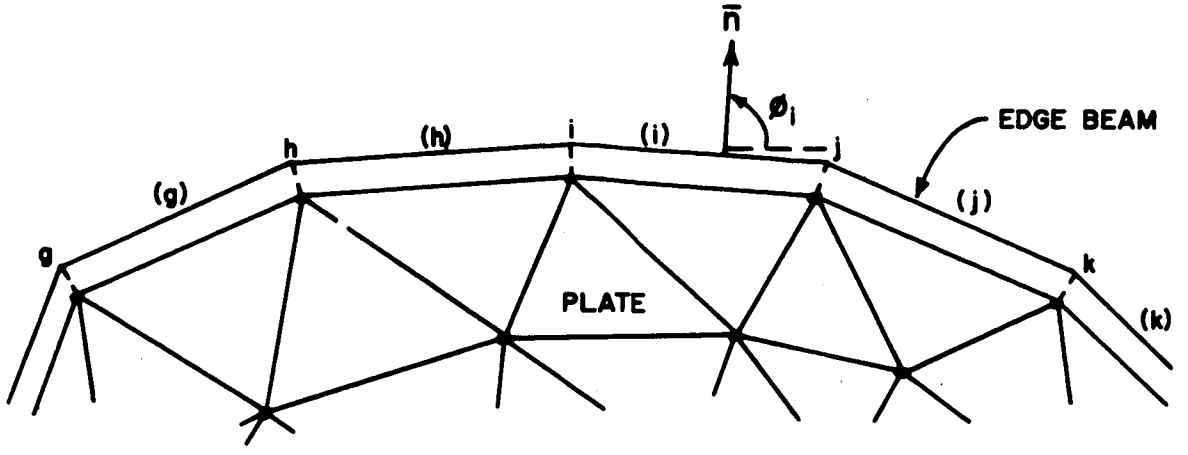


FIGURE 8. A TYPICAL PORTION OF AN EDGE BEAM BOUNDARY.

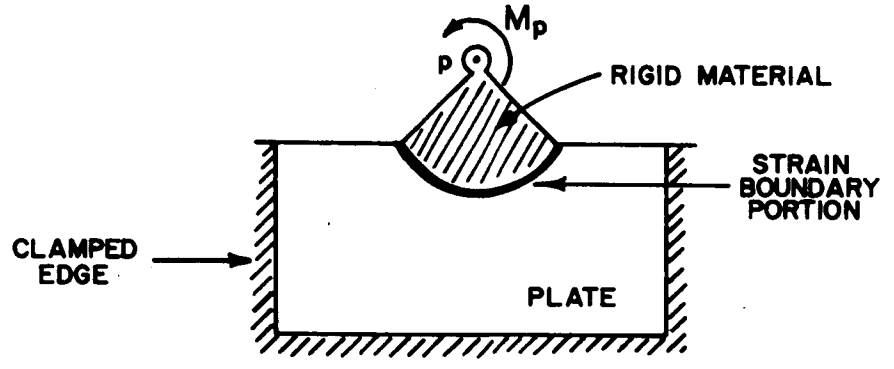


FIGURE 9. EXAMPLE OF A PINNED RIGID BOUNDARY.

and

$$\underset{\sim}{\epsilon}_i = \left\{ \frac{\epsilon_i}{\frac{1}{2}\times_i (\ell_i + \ell_{i-1})} \right\} \tag{95}$$

Eq (93) then replaces Eq (26) for node i in the total system (27).

Specifically the modifications are

$$\left.\begin{aligned}
&\underset{\sim}{K}_{i\ i-1} \Rightarrow -\underset{\sim}{J}_{i-1} && (96)\\
&\underset{\sim}{K}_{ii} \Rightarrow \underset{\sim}{J}_{i-1} + \underset{\sim}{H}_i && (97)\\
&\underset{\sim}{K}_{i\ i+1} \Rightarrow -\underset{\sim}{H}_i && (98)\\
&\underset{\sim}{K}_{ij} \Rightarrow \underset{\sim}{O}_2 && (99)\\
&\underset{\sim}{S}_i \Rightarrow \underset{\sim}{\epsilon}_i && (100)
\end{aligned}\right\} \quad \begin{aligned} i &= \text{all nodes on the boundary portion except the two end nodes}\\ j &= 1,2,\ldots,n\\ j &\neq i-1,i,i+1 \end{aligned}$$

where $\underset{\sim}{O}_2$ is a 2x2 zero matrix.

If the rigid body displacement of the boundary is specified, the two displacement components (given for end node 1) are treated as in the case of displacement boundary conditions. The specified rotation of edge segment (1), ω_1, is combined with Eq (91) for segment (1) to give

$$-\underset{\sim}{M}_1\ \underset{\sim}{U}_2 = \left\{ \frac{\epsilon_1}{\omega_1} \right\} - \underset{\sim}{M}_1\ \underset{\sim}{U}_1 \tag{101}$$

where, in general for side i,

$$\underset{\sim}{M}_i = \frac{1}{\ell_i} \left[\frac{\sin\phi_i}{\cos\phi_i} \middle| \frac{-\cos\phi_i}{\sin\phi_i} \right] \tag{102}$$

which is then used to replace super-row s of $\underset{\sim}{K}$ and $\underset{\sim}{S}$ and results in the following modifications:

$$\underset{\sim}{K}_{s2} \Rightarrow -\underset{\sim}{M}_1 \tag{103}$$

$$\underset{\sim}{K}_{sj} \Rightarrow \underset{\sim}{O}_2 \quad (j = 3,4,\ldots,n) \tag{104}$$

$$\underset{\sim}{S}_j \Rightarrow \left\{ \frac{\epsilon_1}{\omega_1} \right\} - \underset{\sim}{M}_1\ \underset{\sim}{U}_1 \tag{105}$$

If no rigid body motion of the boundary portion is specified, two force resultants and one moment resultant of the boundary forces must be specified for the boundary portion. In matrix form, the force resultants of the boundary portion forces are given by

$$\sum_{i=1}^{s} (\underset{\sim}{R}_i) = \underset{\sim}{\bar{F}} \tag{106}$$

where $\underset{\sim}{\bar{F}} = \left\{ \bar{F}_x \quad \bar{F}_y \right\}$ is the matrix of forces applied externally to the rigid boundary portion. If $\underset{\sim}{\zeta}_i$ is used to represent the (2x2n) supermatrix of super-row i of the stiffness matrix,

$$\underset{\sim}{\zeta}_i = \left[\underset{\sim}{K}_{i1} \quad \underset{\sim}{K}_{i2} \quad \cdots \quad \underset{\sim}{K}_{ii} \quad \cdots \quad \underset{\sim}{K}_{in} \right] \tag{107}$$

then Eqs (106) become, upon use of Eqs (26),

$$\left(\sum_{i=1}^{s} \underset{\sim}{\zeta}_i \right) \underset{\sim}{U} = \sum_{i=1}^{s} (\underset{\sim}{S}_i) + \underset{\sim}{\bar{F}} \tag{108}$$

Eqs (108) are written at node 1 and result in the following changes in super-row 1 of $\underset{\sim}{K}$ and $\underset{\sim}{S}$:

$$\underset{\sim}{K}_{1j} \Rightarrow \sum_{i=1}^{s} \underset{\sim}{K}_{ij} \qquad (j=1,2,\ldots,n) \tag{109}$$

$$\underset{\sim}{S}_1 \Rightarrow \underset{\sim}{\bar{F}} + \sum_{i=1}^{s} (\underset{\sim}{S}_i) \tag{110}$$

The moment resultant of the boundary portion forces about a specified point p with coordinates x_p and y_p is given by

$$\sum_{i=1}^{s} (\underset{\sim}{R}_i)^T \left\{ \begin{matrix} y_p - y_i \\ x_i - x_p \end{matrix} \right\} = \underset{\sim}{\bar{M}}_p \tag{111}$$

The moment resultant is combined with Eq (91) written for segment (1) and the resulting pair of equations is used as the pair of equations for node s, thus replacing super-row s of Eqs (27). The changes to super-row of $\underset{\sim}{K}$ and $\underset{\sim}{S}$

are as follows:

$$\underset{\sim}{K}_{s1} \Rightarrow \sum_{i=1}^{s} \underset{\sim}{\times}_i \underset{\sim}{K}_{i1} + \underset{\sim}{\Phi}_1 \tag{112}$$

$$\underset{\sim}{K}_{s2} \Rightarrow \sum_{i=1}^{s} \underset{\sim}{\times}_i \underset{\sim}{K}_{i2} - \underset{\sim}{\Phi}_1 \tag{113}$$

$$\underset{\sim}{K}_{sj} \Rightarrow \sum_{i=1}^{s} \underset{\sim}{\times}_i \underset{\sim}{K}_{ij} \qquad (j=3,4,\ldots,n) \tag{114}$$

$$\underset{\sim}{S}_s \Rightarrow \left\{ \frac{\sum_{i=1}^{s} \left[y_p - y_i \;\vdots\; x_i - x_p \right] \underset{\sim}{S}_i + \underset{\sim}{\bar{M}}_p}{\epsilon_1} \right\} \tag{115}$$

where

$$\underset{\sim}{\times}_i = \left[\begin{array}{c|c} y_1 - y_i & x_i - x_1 \\ \hline 0 & 0 \end{array} \right] , \qquad \underset{\sim}{\Phi}_i = \left[\begin{array}{c|c} 0 & 0 \\ \hline \sin\phi_i & -\cos\phi_i \end{array} \right] \tag{116}$$

Finally, if the rigid boundary portion is pinned at some point p not necessarily on the boundary edge and has freedom to rotate about that point, equations relating the displacements of node 1 and the rotation of segment (1) to the known displacements of pin-support point p, $\underset{\sim}{U}_p$, must be used at node 1. The above combination of moment resultant and Eq (91) is still used at node s. At node 1, the equations become

$$\underset{\sim}{R}^T \underset{\sim}{U}_1 + \frac{X_p}{\ell_1} \left[\begin{array}{c|c} \cos\phi_1 & \sin\phi_1 \\ \hline 0 & 0 \end{array} \right] (\underset{\sim}{U}_1 - \underset{\sim}{U}_2) = \underset{\sim}{R}^T \underset{\sim}{U}_p \tag{117}$$

where $\underset{\sim}{R}$ is given by Eq (58) if θ replaces ϕ_i. Angle θ relates the line joining point p and node 1 to the global reference frame. X_p is the length of that line (see Fig. 11). The changes to $\underset{\sim}{K}$ and $\underset{\sim}{S}$ required to treat the pinned rigid boundary are as follows:

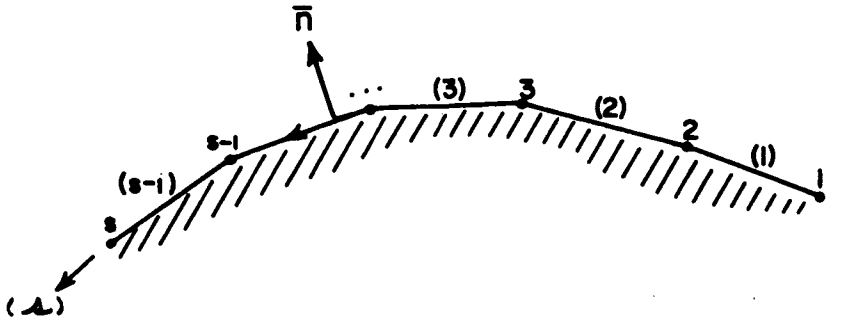


FIGURE 10. NOTATION FOR THE STRAIN BOUNDARY PORTION.

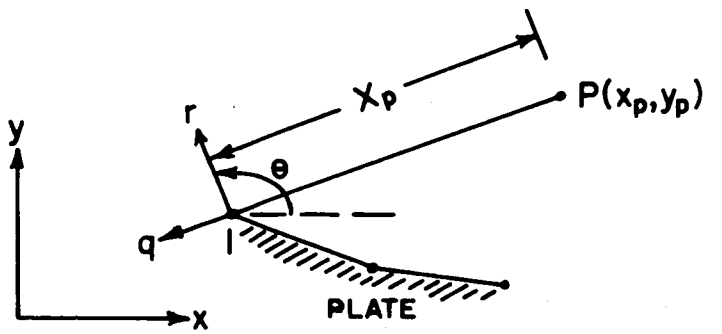


FIGURE 11. NOTATION FOR THE PINNED RIGID BOUNDARY.

$$\underset{\sim}{K}_{11} \Rightarrow \underset{\sim}{R}^T \underset{\sim}{U}_1 + \frac{X_p}{\ell_1} \left[\begin{array}{c|c} \cos\phi_1 & \sin\phi_1 \\ \hline 0 & 0 \end{array}\right] \tag{118}$$

$$\underset{\sim}{K}_{12} \Rightarrow \frac{-X_p}{\ell_1} \left[\begin{array}{c|c} \cos\phi_1 & \sin\phi_1 \\ \hline 0 & 0 \end{array}\right] \tag{119}$$

$$\underset{\sim}{K}_{1j} \Rightarrow \underset{\sim}{Q}_2 \qquad (j=3,4,\ldots,n) \tag{120}$$

$$\underset{\sim}{S}_1 \Rightarrow \underset{\sim}{R}^T \underset{\sim}{U}_p \tag{121}$$

7) Dislocations in Multiply Connected Plates:

A multiply connected plate (one with more than one closed boundary curve) may have internal, self-equilibrating stresses corresponding to the closing of dislocations (Refs. 7 and 11). These initial stresses are continuous across the closed dislocation but the displacements of each node k along the dislocation are multivalued. Let $\underset{\sim}{U}_k^+$ be the matrix displacement of node k on the positive side of the dislocation and $\underset{\sim}{U}_k^-$ be the matrix displacement on the negative side (Fig. 12). The difference between them is related to the rigid-body motion of the positive side of the dislocation in the closing of the dislocation:

$$\delta \underset{\sim}{U}_k = \underset{\sim}{U}_k^+ - \underset{\sim}{U}_k^- = \left\{\begin{array}{c} \delta u^o \\ \delta v^o \end{array}\right\} + \delta\omega^o \left\{\begin{array}{c} -y_k \\ x_k \end{array}\right\} \tag{122}$$

where δu^o, δv^o are the rigid-body translations and $\delta\omega^o$ is the rigid-body rotation about the global origin. Eq (26) written for a node k on the dislocation includes an additional generalized nodal force representing the initial stresses due to the closing of the dislocation in the form

$$\delta \underset{\sim}{F}_k = \frac{1}{2} \left\{ (\underset{\sim}{K}^m_{ki} - \underset{\sim}{K}^s_{ki}) \underset{\sim}{U}_i + \left(\sum_n \underset{\sim}{K}^n_{kk} - \sum_p \underset{\sim}{K}^p_{kk} \right) \underset{\sim}{U}_k + (\underset{\sim}{K}^e_{kj} - \underset{\sim}{K}^d_{kj}) \underset{\sim}{U}_j \right\} \tag{123}$$

where nodes i and j, and elements m, s, e, and d are located as shown in Fig. 12, and

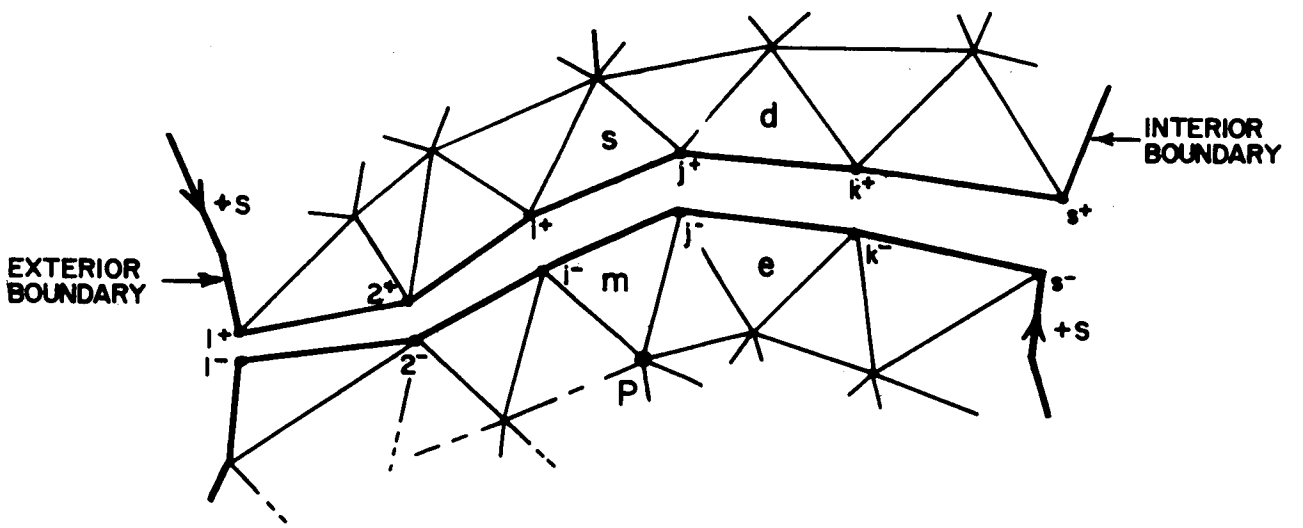


FIGURE 12. NOTATION FOR THE DISLOCATION PROBLEM.

n = all elements incident upon node k on the negative side of the dislocation (124)

p = all elements incident upon node k on the positive side of the dislocation (125)

For a node p not on the dislocation but connected to it by an element, Eq (26) will involve the displacements of at least one dislocation node, say q. One of the terms $\underset{\sim}{K}_{pq}\underset{\sim}{U}_q^-$ or $\underset{\sim}{K}_{pq}\underset{\sim}{U}_q^+$ will appear in Eq (26) depending on which side of the dislocation p is located. The "average" displacement at q can be expressed as

$$\underset{\sim}{U}_q = \underset{\sim}{U}_q^- + \tfrac{1}{2}\underset{\sim}{\delta U}_q = \underset{\sim}{U}_q^+ - \tfrac{1}{2}\underset{\sim}{\delta U}_q \tag{126}$$

so that

$$\underset{\sim}{K}_{pq}\underset{\sim}{U}_q^- = \underset{\sim}{K}_{pq}\underset{\sim}{U}_q + \tfrac{1}{2}\underset{\sim}{K}_{pq}\underset{\sim}{\delta U}_q \tag{127}$$

$$\underset{\sim}{K}_{pq}\underset{\sim}{U}_q^+ = \underset{\sim}{K}_{pq}\underset{\sim}{U}_q - \tfrac{1}{2}\underset{\sim}{K}_{pq}\underset{\sim}{\delta U}_q \tag{128}$$

Since the second term in each of Eqs (127) and (128) is completely known, it can be moved to the right-hand side of Eq (26) and treated as an additional generalized nodal load $\underset{\sim}{\delta F}_p$.

In summary, the modifications to the generalized nodal load matrix $\underset{\sim}{S}$ required to include the closing of dislocations are as follows:

a) for each node i lying on the dislocation:

$$\underset{\sim}{S}_i \Rightarrow \underset{\sim}{S}_i + \underset{\sim}{\delta F}_i \quad (\underset{\sim}{\delta F}_i \text{ given by Eq (123)}) \tag{129}$$

b) for each node p connected to the dislocation on the positive side:

$$\underset{\sim}{S}_p \Rightarrow \underset{\sim}{S}_p - \tfrac{1}{2}\sum_q \underset{\sim}{K}_{pq}\underset{\sim}{\delta U}_q \tag{130}$$

c) for each node p connected to the dislocation on the negative side:

$$\underset{\sim}{S}_p \Rightarrow \underset{\sim}{S}_p + \tfrac{1}{2}\sum_q \underset{\sim}{K}_{pq}\underset{\sim}{\delta U}_q \tag{131}$$

where in both b) and c) the summation q extends over all nodes that are on the dislocation and connected to p.

3.2 Introduction of Boundary Conditions for the Dual Plate Bending Problem

1) Displacement Boundary Conditions:

Displacement boundary conditions take the form of the z component of nodal displacement (w) being known along a portion of the boundary along with slope of the plate edge in the direction perpendicular to the edge ($w_{,n}$). In the finite element method, only the average value of $w_{,n}$ over an edge segment need be known. The average value of $w_{,s}$ (the slope of the edge in the direction parallel to the edge) is computed for each segment i from

$$(w_{,s})_i = \left[(w)_{i+1} - (w)_i \right] / \ell_i \tag{132}$$

where nodes i and i+1 are at each end of segment i. The cartesian components of average edge slope for segment i then become

$$w_{,x} = w_{,n} \cos\phi_i - w_{,s} \sin\phi_i \tag{133}$$

$$w_{,y} = w_{,n} \sin\phi_i - w_{,s} \cos\phi_i \tag{134}$$

Now consider a node k on the boundary, between segments m and n. The total generalized nodal rotations due to edge slope can be computed directly from $w_{,x}$ and $w_{,y}$ (Ref. 7).

$$R'_{xk} = (w_{,n})_n \sin\phi_n + (w_{,s})_n \cos\phi_n - (w_{,n})_m \sin\phi_m - (w_{,s})_m \cos\phi_m \tag{135}$$

$$R'_{yk} = (w_{,n})_n \cos\phi_n + (w_{,s})_n \sin\phi_n + (w_{,n})_m \cos\phi_m - (w_{,s})_m \sin\phi_m \tag{136}$$

The portions of these generalized nodal rotation components due to the curvatures of the particular solution are computed from the particular solution functions (Ref. 7). The remaining homogeneous portions are then dual of the generalized nodal forces due to edge loads in the stretching problem. If the matrix of particular solution generalized nodal rotations at node k is $\underset{\sim}{R}'^{p}_{k}$, and that of total generalized nodal rotations due to edge slope is $\underset{\sim}{R}'_k = \left\{ R'_{xk} \quad R'_{yk} \right\}$, then the modifications to $\underset{\sim}{S}'$ for each node k along the portion of the boundary on which displacement conditions are specified are as follows:

$$\underset{\sim}{S}_k \Rightarrow \underset{\sim}{S}_k + \underset{\sim}{R}'_k - \underset{\sim}{R}'^{p}_k \tag{137}$$

2) Stress Boundary Conditions:

Stress boundary conditions take the form of the edge stress couple (M_{nn}) and z component of edge effective shear (Q_{ne}) being known on a portion or portions of the boundary. From these values and the particular solution, obtain

$$M^*_{nn} = M_{nn} - M^p_{nn} \tag{138}$$

$$Q^*_{ne} = Q_{ne} - Q^p_{ne} \tag{139}$$

M^*_{nn} and Q^*_{ne} are dual of ϵ_{ss} and $\times$, respectively, in the stretching problem, and thus stress boundary conditions of the bending problem can be treated exactly as strain boundary conditions of the stretching problem (Eqs (96)-(100)). For a simply-connected plate with stress boundary conditions on only one continuous portion of the boundary, U_1, V_1, and Ω_1, given by an equation dual of (78), can be specified arbitrarily at node 1 (the first node) of the portion. These quantities are dual of the rigid-body motion of the strain boundary portion of the stretching problem, and can be introduced via equations dual of Eqs (52)-(56) for node 1 and dual of Eqs (103)-(105) for node s (the last node on the portion).

For the case of stress boundary conditions specified on more than one portion of the boundary, a more careful, detailed study of the duality properties (Sec. 9. c. of Ref. 7) is required. The results of such study reveal that conditions dual of the force and moment resultant of the strain portions are used when the stress condition portions alternate with displacement conditions. For a typical stress boundary condition portion, the quantities dual of the externally applied force and moment in the stretching problem are

$$\underset{\sim}{\overline{F}}' = \left\{ \overline{F}'_x \quad \overline{F}'_y \right\} = \left\{ \begin{matrix} w^B_{,y} - w^A_{,y} \\ -w^B_{,x} + w^A_{,x} \end{matrix} \right\} \tag{140}$$

$$\overline{M}'_A = w^A - w^B - (x_A - x_B) w^B_{,x} - (y_A - y_B) w^B_{,y} \tag{141}$$

where A and B are the segments immediately preceeding and following the stress portion, and x_A, y_A, x_B, y_B are the coordinates of the midpoints of these segments. Since displacement boundary conditions are given on these two segments, $\underline{F}'$ and M' are completely known.. The resulting modifications to $\underline{K}'$ and $\underline{S}'$ are dual of Eqs (109), (110), (112), (113), (114), and (115). U_1, V_1, and Ω_1 are still prescribed arbitrarily on one of the stress boundary portions.

3) Mixed Boundary Conditions:

a) M_{nn} and w specified: For each segment, an equation dual of Eq (91) can be written relating M^*_{nn} to the nodal values of the stress functions. M^*_{nn} is related to the specified value of M_{nn} by Eq (138).

From Ref. are obtained also the following relations:

$$(x_B - x_A)\, w^B_{,x} + (y_B - y_A)\, w^B_{,y} = w^B - w^A \tag{142}$$

$$w^B_{,x} - w^A_{,x} = -R'_{yk} \tag{143}$$

$$w^B - w^A - (x_B - x_A)\, w^A_{,x} - (y_B - y_A)\, w^A_{,y} = (x_k - x_B)\, R'_{yk} - (y_k - y_B)\, R'_{xk} \tag{144}$$

where A and B are midpoints of two adjacent boundary segments and k is the node between them. Using Eqs (133), (134), and (142) to solve for $w^B_{,n}$, obtain

$$w^B_{,n} = \frac{w^B - w^A + w^B_{,s}\left[\sin\phi_B\,(x_B - x_A) - \cos\phi_B\,(y_B - y_A)\right]}{\cos\phi_B\,(x_B - x_A) + \sin\phi_B\,(y_B - y_A)} \tag{145}$$

This result is not valid for $\phi_A = \phi_B = 0^\circ$, 90°, 180°, 270°. These exceptions will be treated later. Using Eqs (133), (134), and (145) in Eq (143), obtain

$$w^A_{,n} = \frac{\cos\phi_B}{\cos\phi_A}\left\{\frac{w^B - w^A + w^B_{,s}\left[\sin\phi_B\,(x_B - x_A) - \cos\phi_B\,(y_B - y_A)\right]}{\cos\phi_B\,(x_B - x_A) + \sin\phi_B\,(y_B - y_A)}\right\} \tag{146}$$

$$- w'^{B}_{,s} \frac{\sin\phi_B}{\cos\phi_A} + w'^{A}_{,s} \tan\phi_A + \frac{R'_{yk}}{\cos\phi_A} \qquad \text{(146 cont)}$$

Finally, if nodes j and m are the nodes at the other ends of segments A and B, respectively, and $w_{,s}$ is expressed in terms of nodal values of w, then Eq (146) substituted into Eq (144) yields

$$R'_{xk} \cos\phi_A (y_m - y_k) + R'_{yk} \left[\cos\phi_A (x_k - x_j) + \sin\phi_A (y_m - y_j)\right] = \mathcal{F}_k(w) \qquad (147)$$

where $\mathcal{F}_k(w)$ is a function of known nodal displacements and geometric properties, given by

$$\mathcal{F}_k(w) = (y_m - y_j) \left\{ \frac{(w_m - w_j)(\sin\phi_B \cos\phi_A - \sin\phi_A \cos\phi_B)}{(y_m - y_j)\sin\phi_B + (x_m - x_j)\cos\phi_B} \right. \qquad (148)$$

$$\left. - \frac{(w_m - w_k)\left[(x_m - x_j)(\sin\phi_A \sin\phi_B \cos\phi_B - \cos\phi_A) - (y_m - y_j)\cos^2\phi_B \cos\phi_A\right]}{\ell_B \left[(y_m - y_j)\sin\phi_B + (x_m - x_j)\cos\phi_B\right]} + \frac{w_k - w_j}{\ell_A} \right\}$$

Eq (147) is the second equation to be written for each node along the portion, together with an equation dual of Eq (91). Eq (147) can be expressed in terms of the unknown nodal values of the stress functions. The resulting matrix equation for each node along the mixed boundary conditions portion is

$$\sum_{i=1}^{j} \underset{\sim}{w}_k \underset{\sim}{K}'_{ik} \underset{\sim}{U}'_i + (\underset{\sim}{w}_k \underset{\sim}{K}'_{kk} + \underset{\sim}{\Phi}_k) \underset{\sim}{U}'_k + (\underset{\sim}{w}_k \underset{\sim}{K}'_{km} - \underset{\sim}{\Phi}_k) \underset{\sim}{U}'_m$$

$$+ \sum_{i=m+1}^{n} (\underset{\sim}{w}_k \underset{\sim}{K}'_{ik}) \underset{\sim}{U}'_i = \left\{ \frac{\mathcal{F}_k(w)}{\ell_B (M_{nn} - M^p_{nn})_B} \right\} \qquad (149)$$

$$+ \underset{\sim}{w}_k (\underset{\sim}{P}'_k + \underset{\sim}{\theta}'_k - \underset{\sim}{R}'^{p}_k)$$

where

$$\underset{\sim}{w}_k = \left[\begin{array}{c|c} \cos\phi_A (y_m - y_k) & \cos\phi_A (x_k - x_j) + \sin\phi_A (y_m - y_j) \\ \hline 0 & 0 \end{array}\right] \tag{150}$$

The exceptions mentioned above will now be treated.

for $\phi_A = \phi_B = 0$ or 180^o, obtain directly

$$\left.\begin{aligned} w^A_{,y} &= \pm w^A_{,s} \\ w^B_{,y} &= \pm w^B_{,s} \end{aligned}\right\} \quad \begin{array}{l} +\text{for } 0^o \\ -\text{for } 180^o \end{array} \tag{151, 152}$$

Thus, Eqs (145) and (146) can be bypassed and Eq (144) becomes directly

$$w^B - w^A - (y_B - y_A)\, w^A_{,s} = -(y_k - y_B)\, R'_{xk} \tag{153}$$

The resulting simpler expressions for $\mathcal{F}_k(w)$ and $\underset{\sim}{W}_k$ are

$$\mathcal{F}_k(w) = w_m - w_j - \frac{(y_m - y_j)(w_k - w_j)}{A} \tag{154}$$

$$\underset{\sim}{W}_k = \left[\begin{array}{c|c} y_m - y_k & 0 \\ \hline 0 & 0 \end{array}\right] \tag{155}$$

For $\phi_A = \phi_B = 90^o$ or 270^o, obtain directly

$$\left.\begin{aligned} w^A_{,x} &= \pm w^A_{,s} \\ w^B_{,x} &= \pm w^B_{,s} \end{aligned}\right\} \quad \begin{array}{l} +\text{for } 270^o \\ -\text{for } 90^o \end{array} \tag{156, 157}$$

so that Eq (144) becomes

$$w^B - w^A - (x_B - x_A)\, w^A_{,s} = (x_k - x_B)\, R'_{yk} \tag{158}$$

The resulting simpler expressions for $\mathcal{F}_k(w)$ and w_k are

$$\mathcal{F}_k(w) = w_m - w_j - \frac{(x_m - x_j)(w_k - w_j)}{A} \tag{159}$$

$$\underset{\sim}{w}_k = \left[\begin{array}{c|c} 0 & x_k - x_m \\ \hline 0 & 0 \end{array}\right] \tag{160}$$

Finally, the modifications to $\underset{\sim}{K}'$ and $\underset{\sim}{S}'$ necessary for introduction of mixed boundary conditions of the first type are, for each node k on the boundary portion:

$$\underset{\sim}{K}'_{ik} \Rightarrow \underset{\sim}{W}_k \underset{\sim}{K}'_{ik} \qquad (i = 1,2,\ldots j) \tag{161}$$

$$\underset{\sim}{K}'_{kk} \Rightarrow \underset{\sim}{W}_k \underset{\sim}{K}'_{kk} + \underset{\sim}{\Phi}_k \tag{162}$$

$$\underset{\sim}{K}'_{km} \Rightarrow \underset{\sim}{W}_k \underset{\sim}{K}'_{km} - \underset{\sim}{\Phi}_k \tag{163}$$

$$\underset{\sim}{K}'_{ik} \Rightarrow \underset{\sim}{W}_k \underset{\sim}{K}'_{ik} \qquad (i = m+1, m+2, \ldots n) \tag{164}$$

$$\underset{\sim}{S}'_k \Rightarrow \underset{\sim}{W}_k (\underset{\sim}{P}'_k + \underset{\sim}{\Theta}'_k - \underset{\sim}{R}'^{p}_k) + \left\{ \frac{\mathcal{F}_k(w)}{\ell_k (M_{nn} - M^{p}_{nn})_k} \right\} \tag{165}$$

b) $w_{,n}$ and Q_{ne} specified: For each node, an equation dual of Eq (92) can be written relating Q^*_{ne} to the nodal values of the stress functions. Q^*_{ne} is related to the specified value of Q_{ne} by Eq (139).

From Ref. are obtained the following relations:

$$w^k_{,x} - w^f_{,x} = - \sum_{i=f+1}^{k} R'_{yi} \tag{166}$$

$$w^k_{,y} - w^f_{,y} = \sum_{i=f+1}^{k} R'_{xi} \tag{167}$$

where $w^j_{,x}$ or $w^j_{,y}$ is an average value over segment j following node j. The nodes of the portion are numbered consecutively from 1 to s. In addition, singlevaluedness conditions for $w_{,x}$, $w_{,y}$ and w are obtained from Eqs (166) and (167) and from Eq (189) of Ref. 7.

$$\sum_k R'_{yk} = 0 \tag{168}$$

$$\sum_k R'_{xk} = 0 \tag{169}$$

$$\sum_k \left\{ (x_k - x_p) R'_{yk} - (y_k - y_p) R'_{xk} \right\} = 0 \tag{170}$$

where the symbol $\sum$ means "sum on all nodes around the entire closed boundary curve," and p in any point on the boundary. And finally, there is the relation

$$w'^{i}_{,n} = w'^{i}_{,x} \cos\phi_i + w'^{i}_{,y} \sin\phi_i \tag{171}$$

for all segments i along the mixed boundary condition portion.

Let q be the number of the first segment along the portion for which $\phi_q \neq 90^\circ$ or 270° (segment numbers correspond to the numbers of the nodes they follow.) Let r be the first segment <u>after</u> segment q for which $\phi_r \neq \phi_q$. This somewhat strange procedure is necessary because Eqs. (166) and (167) must be written between two segments that are not parallel to the x-axis. Only in this way will Eqs (166) and (167) eventually yield values of the two additional unknowns $w'^{q}_{,x}$ and $w'^{q}_{,y}$. Writing Eqs (166) and (167) between segments q and r, obtain

$$w'^{r}_{,x} - w'^{q}_{,x} = - \sum_{i=q+1}^{r} R'_{yi} \tag{172}$$

$$w'^{r}_{,y} - w'^{q}_{,y} = \sum_{i=q+1}^{r} R'_{xi} \tag{173}$$

Applying Eq (171) to segments q and r, and using Eqs (172) and (173), the quantities $w'^{q}_{,x}$ and $w'^{q}_{,y}$ are obtained as

$$w^q_{,x} = \frac{w^q_{,n}}{\cos\phi_q} - \mathrm{Tan}\phi_q \, w^q_{,y} \tag{174}$$

$$w^q_{,y} = \frac{w^r_{,n}\cos\phi_q - w^q_{,n}\cos\phi_r + \cos\phi_q\left(\cos\phi_r \sum_{i=q+1}^{r} R'_{yi} - \sin\phi_r \sum_{i=q+1}^{r} R'_{xi}\right)}{\sin(\phi_r - \phi_q)} \tag{175}$$

Finally, the following equations relating the stress functions to known quantities may be written for each node along the boundary portion:

i) for node j such that $1 < j < q$:

$$w^{j-1}_{,n} = \cos\phi_{j-1}\left(w^q_{,x} + \sum_{i=j}^{q} R'_{yi}\right) + \sin\phi_{j-1}\left(w^q_{,y} - \sum_{i=j}^{q} R'_{xi}\right) \tag{176}$$

ii) for done j such that $q < j \leq s$, <u>except</u> node r:

$$w^j_{,n} = \cos\phi_j\left(w^q_{,x} - \sum_{i=q+1}^{j} R'_{yi}\right) + \sin\phi_j\left(w^q_{,y} + \sum_{i=q+1}^{j} R'_{xi}\right) \tag{177}$$

iii) for node r, Eq (168) expressed in terms of the stress functions, the particular solution, and thermal nodal rotations, if any.

To specifiy the matrix modifications necessary for the introduction of the above equations and the dual of Eq (92), the following matrices are defined:

$$\underset{\sim}{C}_j = \left[\begin{array}{c|c} -\sin\phi_j & \cos\phi_j \\ \hline 0 & 0 \end{array}\right] \tag{178}$$

$$\underset{\sim}{Q} = \frac{1}{\sin(\phi_r - \phi_q)}\left[\begin{array}{c|c} \sin\phi_q \sin\phi_r & -\sin\phi_q\cos\phi_r \\ \hline -\cos\phi_q\sin\phi_r & \cos\phi_q\cos\phi_r \end{array}\right] \tag{179}$$

$$\underset{\sim}{Q}_i = \underset{\sim}{\theta}'_i + \underset{\sim}{P}'_i - \underset{\sim}{R}'^{p}_i \tag{180}$$

and the scalar quantity $\mathcal{W}_j$ is defined:

$$\mathcal{W}_j = w^j_{,n} - \cos\phi_j \left\{ \frac{w^q_{,n}}{\cos\phi_q} - \frac{\tan\phi_q (w^r_{,n}\cos\phi_q - w^q_{,n}\cos\phi_r)}{\sin(\phi_r - \phi_q)} \right\} - \frac{\sin\phi_j}{\sin(\phi_r - \phi_q)} (w^r_{,n}\cos\phi_q - w^q_{,n}\cos\phi_r) \tag{181}$$

Then the modifications to $\underset{\sim}{K}'$ and $\underset{\sim}{S}'$ are as follows:

i) for row j such that $1<j\leq q$:

$$\underset{\sim}{K}'_{jk} \Rightarrow \underset{\sim}{B}_k \quad (k=1,2,\ldots,n\neq j-1,j,j+1) \tag{182}$$

$$\underset{\sim}{K}'_{j,j-1} \Rightarrow \underset{\sim}{B}_{j-1} - \underset{\sim}{J}_{j-1} \tag{183}$$

$$\underset{\sim}{K}'_{jj} \Rightarrow \underset{\sim}{B}_j + \underset{\sim}{J}_{j-1} + \underset{\sim}{J}_j \tag{184}$$

$$\underset{\sim}{K}'_{j,j+1} \Rightarrow \underset{\sim}{B}_{j+1} - \underset{\sim}{J}_j \tag{185}$$

$$\underset{\sim}{S}'_j \Rightarrow \underset{\sim}{\mathcal{C}}_{j-1} \left\{ \sum_{i=j}^{q} \underset{\sim}{Q}_i + \underset{\sim}{\mathcal{Q}} \sum_{i=q+1}^{r} \underset{\sim}{Q}_i \right\} + \left\{ \frac{\mathcal{W}_{j-1}}{\frac{Q^*_{ne}}{2}(\ell_j + \ell_{j-1})} \right\} \tag{186}$$

where

$$\underset{\sim}{B}_k = \underset{\sim}{\mathcal{C}}_{j-1} \left\{ \sum_{i=j}^{j} \underset{\sim}{K}'_{ik} - \underset{\sim}{\mathcal{Q}} \sum_{i=q+1}^{r} \underset{\sim}{K}'_{ik} \right\} \tag{187}$$

ii) for row j such that $q<j<s$, <u>except</u> row r:

$$\underset{\sim}{K}'_{jk} \Rightarrow \underset{\sim}{A}_k \quad (k=1,2,\ldots,n\neq j-1,j,j+1) \tag{188}$$

$$\underset{\sim}{K}'_{j,j-1} \Rightarrow \underset{\sim}{A}_{j-1} - \underset{\sim}{J}_{j-1} \tag{189}$$

$$\underset{\sim}{K}'_{jj} \Rightarrow \underset{\sim}{A}_j + \underset{\sim}{J}_{j-1} + \underset{\sim}{J}_j \tag{190}$$

$$\underset{\sim}{K}'_{j,j+1} \Rightarrow \underset{\sim}{A}_{j+1} - \underset{\sim}{J}_j \tag{191}$$

$$\underset{\sim}{S}_j \Rightarrow -\underset{\sim}{\mathcal{C}}_j \left\{ \sum_{i=q+1}^{j} \underset{\sim}{Q}_i - \underset{\sim}{\mathcal{Q}} \sum_{i=q+1}^{r} \underset{\sim}{Q}_i \right\} + \left\{ \frac{\mathcal{W}_j}{\frac{Q^*_{ne}}{2}(\ell_j + \ell_{j-1})} \right\} \tag{192}$$

where

$$\underset{\sim}{A}_k = -\underset{\sim}{\mathcal{C}}_j \left\{ \sum_{i=q+1}^{j} \underset{\sim}{K}'_{ik} - \underset{\sim}{\mathcal{Q}} \sum_{i=q+1}^{r} \underset{\sim}{K}'_{ik} \right\} \tag{193}$$

iii) for row r:

$$\underset{\sim}{K}'_{rk} \Rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \sum_i \underset{\sim}{K}'_{ik} \qquad (k=1,2,\ldots,n\neq r-1,r,r+1) \tag{194}$$

$$\underset{\sim}{K}'_{r,r-1} \Rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \sum_i \underset{\sim}{K}'_{i,r-1} - \underset{\sim}{J}_{r-1} \tag{195}$$

$$\underset{\sim}{K}'_{rr} \Rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \sum \underset{\sim}{K}'_{ir} + \underset{\sim}{J}_{r-1} + \underset{\sim}{J}_r \tag{196}$$

$$\underset{\sim}{K}'_{r,r+1} \Rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \sum \underset{\sim}{K}'_{r,r+1} - \underset{\sim}{J}_r \tag{197}$$

$$\underset{\sim}{S}'_r \Rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \sum \underset{\sim}{Q}_i + \left\{ \begin{array}{c} 0 \\ \hline \frac{Q^*_{ne}}{2}(\ell_r + \ell_{r-1}) \end{array} \right\} \tag{198}$$

where, in all the above modifications, $\underset{\sim}{J}_n$ is given by Eq (94).

If no segment along the portion can be found such that $\phi = 90^\circ$ or 270° for that segment, then all segments are parallel to the x-axis and $w_{,n} = \pm w_{,y}$ for each segment. The matrices for the above modification procedure become

$$\left. \begin{array}{l} \underset{\sim}{\mathcal{C}}_j = \begin{bmatrix} \pm 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \underset{\sim}{\mathcal{Q}} = 0 \\ \end{array} \right\} \begin{array}{l} \text{- for } 90^\circ \\ \text{+ for } 270^\circ \end{array} \tag{199}$$

and the scalar $\mathcal{W}_j$ becomes

$$\mathcal{W}_j = w^j_{,n} - w^1_{,n} \tag{200}$$

and q=r=1 in the steps of the procedure.

If a segment q is found but no segment r after q can be found such that $\phi_q \neq \phi_r$, the above simplified matrices and procedure, Eqs (199) and (200), may be used for all nodes j before node q. Since all segments after node q are parallel, i.e., the boundary portion from node q to the end of the mixed boundary condition portion is a straight line, the following simpler equation is used at all nodes j (q<j<s)

$$w^j_{,n} - w^q_{,n} = -\sum_{i=q+1}^{k} R'_{yi} \cos\phi_q + \sum_{i=q+1}^{k} R'_{xi} \sin\phi_q \tag{201}$$

resulting in

$$W_j = w^j_{,n} - w^q_{,n} \tag{202}$$

$$\underset{\sim}{C}_j = \left[\begin{array}{c|c} -\sin\phi_q & \cos\phi_q \\ \hline 0 & 0 \end{array}\right] \qquad \underset{\sim}{Q} = \underset{\sim}{0} \tag{203}$$

The singlevaluedness condition is used at node q, i.e., q=r.

4) Elastic Boundary Conditions:

Elastic boundary conditions for the plate bending problem are dual of an edge beam in the plate stretching problem (Ref. 7). The duality correspondence is given in Table II.

Table II

Stretching	Bending
EA, EI	$-f_{ss}$, $-f_{zz}$ (elastic flexibility coefficients)
ϵ^o	$M^o = M^s - M^p_{nn}$
$\varkappa^o$	$Q^o = Q^s - Q^p_{ne}$
$\bar{N}_x$, $\bar{N}_y$	$-K_x\left(\frac{dy}{ds}\right)$, $K_y\left(\frac{dx}{ds}\right)$

where Q^s and M^s are the edge shear and couple due to support movement, given by

$$Q^s = \frac{w^s}{f_{zz}} \quad , \quad M^s = \frac{w^s_{,n}}{f_{ss}} \tag{204}$$

Thus, the matrix modifications necessary for the treatment of an elastic boundary in the bending problem are dual of Eqs (89) and (90), using Table 2.

5) Plate Bounded by an Edge Beam:

An edge beam boundary for the plate bending problem is the dual of elastic boundary conditions for the stretching problem (Ref. 7). To develop the dual matrix modifications, it is necessary to define the following coefficients (dual of the elastic stiffness constants of the stretching problem elastic boundary):

$$f_{xx} = \frac{(dy/ds)^2}{EI} + \frac{(dx/ds)^2}{GJ} \tag{205}$$

$$f_{xy} = f_{yx} = \frac{dx}{ds}\frac{dy}{ds}\left(\frac{1}{GJ} - \frac{1}{EI}\right) \tag{206}$$

$$f_{yy} = \frac{(dx/ds)^2}{EI} + \frac{(dy/ds)^2}{GJ} \tag{207}$$

and the support stress functions (dual of the support movement)

$$U^s = (U_o + C^b_{xo}) - (y - y_o)(\Omega_o + P^b_{zo}) + C'_x \tag{208}$$

$$V^s = (V_o + C^b_{yo}) - (x - x_o)(\Omega_o + P^b_{zo}) + C'_y \tag{209}$$

where the subscript o refers to an arbitrary initial point on the edge beam boundary, and C'_x and C'_y are the moments at a cross-section of the beam due to the forces $(\bar{Q}_{ne} - Q^p_{ne})$ and moments $(\bar{M}_{nn} - M^p_{nn})$ acting on the beam portion oriented positively from point o to the cross-section. $\bar{Q}_{ne}$ and $\bar{M}_{nn}$ are the edge shear and stress couple applied externally to the beam. Since these stresses are known, along with the particular solution, the quantities U^s and V^s are completely determined except for the three quantities

$$(\Omega_o + P^b_{zo}) \ , \ (U_o + C^b_{xo}) \ , \ (V_o + C^b_{yo}) \tag{210}$$

(See Ref. 7 for a definition and discussion of these quantities.)
For a simply connected plate, U_o, V_o, and Ω_o are arbitrary and thus may be chosen so that the three quantities (210) vanish, leaving

$$U^s = C'_x \tag{211}$$

$$V^s = C'_y \tag{212}$$

For a multiply connected plate, the three quantities (210) still vanish on one of the boundary curves. For the other curves, the three singlevaluedness conditions, Eqs (168), (169), and (170), are used to determine the three quantities (210) by expressing R'_{xk} and R'_{yk} in terms of the flexibility constants, stress functions U and V, and support functions through equations dual of (66) and (67).

In matrix form, Eqs (208) and (209) become

$$\underset{\sim}{U}^s_k = \underset{\sim}{U}^s_o + \underset{\sim}{L}^T_k Z_o + \underset{\sim}{C}'_k \tag{213}$$

where

$$\underset{\sim}{U}^s_k = \left\{ U^s_k \quad V^s_k \right\} \qquad \underset{\sim}{L}_k = \left[y_o - y_k \,\vdots\, x_k - x_o \right] \tag{214}$$

$$\underset{\sim}{U}^s_o = \left\{ \begin{array}{c} U_o + C^b_{xo} \\ \hdashline V_o + C^b_{yo} \end{array} \right\} \qquad \underset{\sim}{C}'_k = \left\{ C'_x \,\vdots\, C'_y \right\}_k \tag{215}$$

$$Z_o = \Omega_o + P^b_{zo} \tag{216}$$

In applying the singlevaluedness conditions to equations dual of (66) and (67), it is convenient to define the following matrices and scalars:

$$\underset{\sim}{\mathcal{F}}_k = (\underset{\sim}{f}_{k-1,k} + \underset{\sim}{f}_{kk} + \underset{\sim}{f}_{k+1,k}) \tag{217}$$

where $\underset{\sim}{f}_{ij}$ is a flexibility matrix dual of k_{ij} and is determined from the flexibility constants through equations dual of (69), (70), (71), and (72);

$$\frac{1}{A} = \sum_k \underset{\sim}{L}_k \underset{\sim}{\mathcal{F}}_k \underset{\sim}{L}^T_k \qquad \text{(a scalar)} \tag{218}$$

$$\underset{\sim}{B}_k = \underset{\sim}{L}_k \underset{\sim}{\mathcal{F}}_k \tag{219}$$

$$\underset{\sim}{H} = \sum_k \underset{\sim}{L}_k \underset{\sim}{\mathcal{F}}_k \tag{220}$$

Using these matrices and scalars, the solutions for the three quantities (210) are obtained as

$$\underset{\sim}{U}_o^s = \left(A \underset{\sim}{H}^T \underset{\sim}{H} - \sum_k \underset{\sim}{\mathcal{F}}_k \right)^{-1} \left\{ \sum_k \left(A \underset{\sim}{H}^T \underset{\sim}{B}_k - \underset{\sim}{\mathcal{F}}_k \right) \underset{\sim}{U}_k + \sum_k \left[\left(A \underset{\sim}{H}^T \sum_k \underset{\sim}{L}_k - \underset{\sim}{I}_2 \right) \underset{\sim}{F}_k'^p + \left(A \underset{\sim}{H}^T \underset{\sim}{B}_k + \underset{\sim}{\mathcal{F}}_k \right) \underset{\sim}{C}_k' \right] \right\} \tag{221}$$

$$Z_o = A \left\{ \sum_k \underset{\sim}{L}_k \underset{\sim}{F}_k'^p + \sum_k \underset{\sim}{B}_k \left(\underset{\sim}{U}_k + \underset{\sim}{C}_k' \right) - \underset{\sim}{H} \underset{\sim}{U}_o^s \right\} \tag{222}$$

The modifications to $\underset{\sim}{K}'$ necessary for the inclusion of edge beam effects are dual of Eqs (73), (74), and (75). For simply connected plates, the modifications to $\underset{\sim}{S}'$ are dual of Eq (76). However, for multiply connected plates, $\underset{\sim}{U}_k^s$ includes terms involving the unknown stress functions U, as is shown by Eqs (221) and (222). These terms must be moved to the left hand side of Eqs (46) and thus lead to additional modifications to $\underset{\sim}{K}'$:

a) for each node i along the edge beam modify row i of $\underset{\sim}{K}'$ as follows

$$\text{to } \underset{\sim}{K}'_{ih} \text{ add } \underset{\sim}{T}_h \tag{223}$$

$$\text{to } \underset{\sim}{K}'_{ii} \text{ add } \underset{\sim}{T}_i \tag{224}$$

$$\text{to } \underset{\sim}{K}'_{ij} \text{ add } \underset{\sim}{T}_j \tag{225}$$

where h and j are the nodes preceeding and following node i along the boundary, and

$$\underset{\sim}{T}_k = \left(A \underset{\sim}{L}_k^T \underset{\sim}{H} - \underset{\sim}{I}_2 \right) \underset{\sim}{D}^{-1} \sum_k \left(A \underset{\sim}{H}^T \underset{\sim}{B}_k - \underset{\sim}{\mathcal{F}}_k \right) - A \underset{\sim}{L}_k^T \sum_k \underset{\sim}{B}_k \tag{226}$$

$$\underset{\sim}{D} = A\underset{\sim}{H}^T\underset{\sim}{H} - \sum_k \underset{\sim}{\mathcal{F}}_k \tag{227}$$

b) for each node i, $\underset{\sim}{S}'_i$ becomes, for multiply connected plates,

$$\underset{\sim}{S}'_i \Rightarrow \underset{\sim}{S}'_i + \underset{\sim}{f}_{ih}\underset{\sim}{\$}_h + \underset{\sim}{f}_{ii}\underset{\sim}{\$}_i + \underset{\sim}{f}_{ij}\underset{\sim}{\$}_j \tag{228}$$

where

$$\underset{\sim}{\$}_k = (\underset{\sim}{I}_2 - A\underset{\sim}{L}_k^T\underset{\sim}{H})\,\underset{\sim}{D}^{-1}\underset{\sim}{G}_{uk} + A\underset{\sim}{L}_k^T\underset{\sim}{G}_{zk} + \underset{\sim}{C}'_k \tag{229}$$

$$\underset{\sim}{G}_{uk} = \underset{\sim}{D}^{-1}\sum_k \left\{(A\underset{\sim}{H}^T\sum_k \underset{\sim}{L}_k - \underset{\sim}{I}_2)\,\underset{\sim}{F}'^p_k + (A\underset{\sim}{H}^T\underset{\sim}{B}_k + \underset{\sim}{\mathcal{F}}_k)\,\underset{\sim}{C}'_k\right\} \tag{230}$$

$$\underset{\sim}{G}_{zk} = A\left\{\sum_k \underset{\sim}{L}_k\underset{\sim}{F}'^p_k + \sum_k \underset{\sim}{B}_k\underset{\sim}{C}'_k\right\} \tag{231}$$

CHAPTER 4

Computer Implementation of the System

4.1 Introduction

The dual finite element for stretching and bending of orthotropic plates must utilize the capabilities of a digital computer to be an advantageous method of solution. To maximize its utility, a dual finite element computer system should be easy to use by an engineer who is not a computer expert. Thus the input to the system should be in the form of a problem-oriented language related to the engineering terminology of the problem. Also, the system should be structured in steps, or modules, that relate to the actual engineering steps used in problem solution. The programs for a particular step should be grouped together so that only the group for the current solution step would necessarily be in the computer at any given time. Finally, the data storage should be flexible so that storage areas would be only as large as required for the current problem step, and only data currently being used would necessarily be in the computer at any given time.

The capabilities for creating a system with the above desirable features are available as part of the Integrated Civil Engineering System (ICES) (Ref. 8 and 9) developed by the MIT Department of Civil Engineering. Specifically, a general Finite Element Analyzer exists (Ref. 1) with the capability of easy modification and sophistication. Thus the dual finite element system of this thesis was programmed as additional capabilities to the Finite Element Analyzer.

The programming of the problem-oriented language input commands was done in Command Definition Language (CDL) (Ref. 9), itself a problem-oriented language. The programming of the problem solving routines was done in ICETRAN (Ref. 9), which is FORTRAN IV (Ref. 10) with the added capability of dynamic memory allocation. The packaging of programs into separate groups, called load modules, and linking these groups together to form a solution procedure is accomplished by a facility called the linkage editor (Ref. 8 and 9).

In the next section, the problem-oriented commands used to solve bending and stretching problems are described in detail. This information is all that is needed by an engineer wishing to use the system to solve specific problems. For those engineers wishing to modify or add to the system, detailed system documentation is given in Appendices 1, 2, 3, and 7. This documentation consists of data structure description, COMMON storage area map, program descriptions and flow charts, and program listings. It is intended to parallel the system documentation of the Finite Element Analyzer, which must also be studied by anyone wishing to modify the system of this thesis.

4.2 Description of the Problem-Oriented Language Commands

The computer capabilities developed in this thesis form a subset of the general capabilities of the Finite Element Analyzer, which in turn is a part of the Structural Design Language (STRUDL). Some familiarity with STRUDL, as described in Reference 12, and a good understanding of the Finite Element Analyzer Language (Chapter 4 of Reference 1) are prerequisites to the use of the capabilities described herein.

Below is a complete outline of the commands that may be used to solve problems of stretching and bending of orthotropic plates by the finite element method. Commands marked with an asterisk (*) are part of the general Finite Element Analyzer and are described in detail on pages 35-46 of Reference 1. Those descriptions will not be repeated here. All new commands developed specifically for the dual orthotropic plate problem are described in detail below.

In the command descriptions, underlined words must appear as shown. Data items can be real (with mandatory decimal point), integer (without decimal point), or alphanumeric (written between single quotation marks). The acceptable modes for each node, element, or boundary name are integer or alphanumeric, and for all other data are real unless stated otherwise.

1. Problem Initiation
   * PROBLEM 'identification' 'title'

2. Unit Declaration
   * UNIT length weight angle temperature time

3. Type Specification
   * TYPE type

   The applicable types for this system of problems are:

   DUAL PLATE STRETCHING symmetry
   DUAL PLATE BENDING symmetry
   DUAL PLATE GENERAL symmetry

The symmetry indicator must be either SYMMETRIC or NONSYMMETRIC depending upon whether the global behavior (stiffness/flexibility) matrix is symmetric or non-symmetric in its final form immediately before solution. This symmetry is dependent upon the type of boundary conditions specified by the user, and is given in the table below.

	SYMMETRIC	NONSYMMETRIC
Stretching problem:	displacement	rigid boundary
	stress	
	mixed	
	elastic	
	edge beam	
	dislocations	
Bending problem:	displacement	stress
	elastic	mixed
		edge beam

If all of the boundary conditions to be specified by the user appear in the SYMMETRIC column, then the used must specify SYMMETRIC in the TYPE command. If any of the boundary conditions to be specified appear in the NONSYMMETRIC column, then the user must specify NONSYMMETRIC.

DUAL PLATE GENERAL must be specified if the plate is to be solved for both stretching and bending.

4. Geometry and Topology

* NODE name COORDINATES X v_x Y v_y Condition

or

* NODE COORDINATES

name X v_x Y v_y Condition

name X v_x Y v_y Condition

.

.

.

In this system of problems, all nodes located on boundaries are indicated by writing BOUNDARY or B as the condition. The condition specification for interior nodes is left blank. These requirements for the condition specification differ from the Finite Element Analyzer, in which only nodes with at least one restrained displacement component are indicated by BOUNDARY or B and all other nodes are FREE, F, or blank.

* ELEMENT INCIDENCES

element name $node_1$ $node_2$ $node_3$

element name $node_1$ $node_2$ $node_3$

.

.

BOUNDARY INCIDENCES

boundary name node name

boundary name node name

.

.

This command is used to assign boundary names to the boundaries of the plate. The name may be an integer or up to eight-character alphanumeric name (enclosed in quotes). Node name may be the name of any node located on the boundary being named. A boundary is defined here as an entire closed path bounding the plate. Thus the number of boundary incidences specified by the user is the connectivity of the plate. This command should not be used until after all node conditions and element incidences have been specified. It must be used before any boundary conditions are specified. The specification of the BOUNDARY INCIDENCES causes the chain of boundary nodes for each boundary to be assembled. Implicit in this assembly is the assumption that no element touches more than one closed boundary curve, so the user must subdivide his plate in such a way that this assumption holds true.

5. Element Property Specification

* ELEMENT list PROPERTIES TYPE 'type' THICKNESS -- v_t EX v_E PX v_{px} -- EY v_{Ey} PY v_{py} CTX v_{cy} G v_G DENS v_D

or

* ELEMENT PROPERTIES

list TYPE 'type' THICKNESS v_t . . . .

list TYPE 'type' THICKNESS v_t . . . .

.

.

where EX = Young's modulus in the global x-direction

EY = Young's modulus in the global y-direction
PX = Poisson's coefficient in the x-direction
PY = Poisson's coefficient in the y-direction
CTX = thermal expansion coefficient in the x-direction
CTY = thermal expansion coefficient in the y-direction
G = shear modulus
DENS = material density of the element
THICKNESS = average thickness of the element
TYPE = the type of element being used, which for this class of problems if 'FTOD', standing for flat triangular orthotropic dual.

If only one direction of an orthotropic property is given (i.e., EX), then the other value is assumed to be the same (i.e., EY = EX) except in the case of PY, which is assumed to be (EX)(PX)/(EY).

The list can be either a single node identifier, a list of them, or a specification of the form N_1 TO N_2 if the group of nodes is named by successive integer numbers with N_1 being the lowest integer and N_2 the highest.

6. Boundary Conditions Specification

The explicit and detailed consideration given to boundary conditions in this thesis has made necessary the development of a distinct boundary condition command with a set of options for explicit specification of the various types of boundary conditions. The general form of this command is as follows:

```
* BOUNDARY CONDITION 'boundary name' type (indicator)
  (additional data items)
  boundary portion  values
  boundary portion  values
  .
  .
  .
```

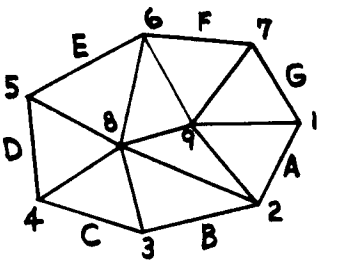
'boundary name' is the name of the boundary to which the particular condition applies. The presence of an indicator and additional data items on the second line, under the tabular heading, depends upon the type of boundary condition being specified.

The beginning of each line under the tabular heading specifies the portion of the given boundary over which the condition applies. This portion may be a single node or a range of nodes

$$\text{node}_1 \ \underline{\text{TO}} \ \text{node}_2$$

The range of nodes implies all segments and nodes along the boundary between node_1 and node_2, traversing the boundary in the +s direction (outward normal vector pointing to the right) from node_1 to node_2. In the right hand x-y-z coordinate system used here, this direction is counter - clockwise for an exterior boundary and clockwise for an interior boundary.

Example:



1 TO 2 would imply nodes 7, 6, 5, 4, and 3, and segments G, F, E, D, and B, whereas

2 TO 1 would imply no additional nodes and only segment A.

1 TO 1 would imply the entire boundary.

The various types of boundary conditions, associated indicators, and data values will now be described.

a) type : DISPLACEMENT Indicator
values : U u V v W w R $w_{,n}$

In the case of a DUAL PLATE GENERAL problem (specifice previously by the TYPE command) involving both stretching and bending, the Indicator is used to tell the system to which problem(s) this particular boundary condition applies, as follows:

STRETCHING (or S)	—	only the stretching problem
BENDING (or B)	—	only the bending problem
GENERAL (or G)	—	both problems

The Indicator is not necessary if the problem type is not DUAL PLATE GENERAL. Value symbols u and v are the x and y components of displacement for all nodes along the specified boundary portion. Symbols w and $w_{,n}$

are the z component of nodal displacement and the negative edge rotation for all nodes and edge segments along the specified portion. All values for a particular portion need not be specified at the same time. Values not specified will be taken as zero unless specified previously or subsequently by another use of the same command.

b) type : STRESS Indicator
values : NX N_{nx} NY N_{ny} Q Q_{ne} MN M_{nn}

The Indicator has the same function as in the DISPLACEMENT type discussed above. Value symbols N_{nx} and N_{ny} are the x and y components of the edge stress resultant (force/unit length) along the boundary portion. The edge stress resultant components are assumed to vary linearly between nodes and thus the specified values apply to all nodes along the boundary portion, including the end nodes. Symbol Q_{ne} is the z component of edge stress resultant (also known as the effective shear) along the boundary portion. Symbol M_{nn} is the edge stress couple (bending moment/unit length) whose vector is oriented in the +s direction. A positive M_{nn} corresponds to a negative $w,_n$. The stress couple M_{nn} is assumed to be constant between two nodes and thus the specified value of M_{nn} applies to all segments along the boundary portion. Again, all values need not be specified at the same time.

c) type : MIXED STRETCHING
values : UR u_r NR N_{nr} ANGLE θ_r

d) type : MIXED BEND/1
values : W w M M_{nn}

e) type : MIXED BEND/2
values : Q Q_{ne} R $w,_n$

Value symbol u_r is the nodal displacement in the direction r (in the plane of the plate), symbol N_{nr} is the edge stress resultant in the direction perpendicular to r (90^o ahead of r) and θ_r is the positive angle from the x-axis to the r-axis. Symbols w, M_{nn}, Q_{ne}, and $w,_n$ are the same as explained above.

f) type : RIGID FIXED
type : RIGID PINNED
type : RIGID FREE
values : CHI $\times_{zs}$ EPSILON ϵ_{ss}

These types of the BOUNDARY CONDITION command are used to specify fixed strain (ϵ_{ss}) and in-plane curvature ($\times_{zs}$) along a portion of the boundary. This is equivalent to making a portion of the boundary rigid with respect to the material of the plate, with a given shape specified by the strain and curvature values; hence the type RIGID. The three alternate forms of the RIGID type state whether the rigid boundary's motion is fixed, pinned, or free with respect to the global coordinates and give the necessary accompanying information on the line immediately below the tabular heading BOUNDARY CONDITION, as follows:

fixed : U1 u_1 V1 v_1 R1 ω

Symbols u_1 and v_1 are the x and y components of imposed displacement of the first node ($node_1$) of the rigid boundary. Symbol ω is the rotation of the edge segment connecting $node_1$ to the next node in the +s direction.

pinned : XP x_p YP y_p UP u_p VP v_p MP M_p

Symbols x_p and y_p are the x and y coordinates (global) of the pinned point p (not necessarily any of the nodes) about which the rigid boundary is free to pivot. Symbols u_p and v_p are the imposed displacements of the pinned point. Symbol M_p is the moment applied externally to the rigid boundary about point p. M_p, as a vector, is parallel to the +z axis of the right-handed x-y-z coordinate system.

free : XP x_p YP y_p FX F_x FY F_y MP M_p

Symbols F_x and F_y are the x and y components of force applied externally to the rigid boundary. The other symbols are the same as for the pinned case.

The RIGID type of boundary condition applies only to the plate stretching problem and any values not specified will be taken as zero.

Examples:

BOUNDARY CONDITION 'OUTER' RIGID PINNED

XP 31. YP 2.7 UP .02 VP .12 MP 220.
'CORNER1' TO 'CORNER2'

This example specifies a pinned rigid boundary with no strain on in-plane curvature, i.e., it has the shape of the plate boundary before deformation.

BOUNDARY CONDITION 'INNER' RIGID FREE
XP 0. YP 0. FX 0. FY 0. MP 0.
21 TO 21

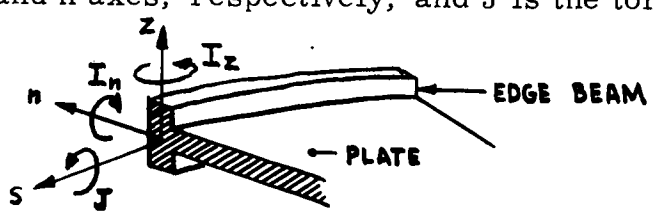
This example could be used to specify a rigid, unloaded "plug" inserted into the entire hole bounded by 'INNER'.

g) type : ELASTIC
values : US u^s VS v^s KXX k_{xx} KXY k_{xy} KYX k_{yx} KYY k_{yy}
WS w^s RS $w^s_{,n}$ KZZ k_{zz} KSS k_{ss}

This type specifies that the entire boundary 'boundary name' is elastically supported. Symbols k_{xx}, k_{xy}, k_{yx}, k_{yy}, $k_{zz} = 1/f_{zz}$, and $k_{ss} = 1/f_{ss}$ are the elastic constants over the specified boundary portions. Symbols u^s, v^s, w^s, and $w^s_{,n}$ are the support movements (3 components of displacement and an edge rotation) along the specified boundary portions. Only those values pertaining to the problem to be solved need be specified. If the sum of all specified boundary portions does not add up to the entire boundary, portions not specified will assume zero values of all relevant data.

h) type : EDGE BEAM
values : NX N_{nx} NY N_{ny} EB E IZ I_z M M_{nn} Q Q_{ne} IN I_n J J G G

This type specifies that the entire boundary 'boundary name' is an edge beam. Symbols E and G are the Young's modulus and shear modulus of the beam material. Symbols I_z and I_n are the moments of inertia of the edge beam about the z and n axes, respectively, and J is the torsional constant, as shown below:



Symbols N_{nx} and N_{ny} are the x and y components of the stress resultant (force/unit length) applied externally to the beam, Q_{ne} is the effective shear applied externally, and M_{nn} is the externally-applied edge moment intensity. Again, only pertinent values need be specified and all values along portions not specified will assume zero values.

In addition to the above boundary condition command, there are a number of so-called standard boundaries that may be more easily specified with the following commands:

SIMPLE SUPPORT 'boundary name' $node_1$ TO $node_2$

This support is the plate equivalent of a pin-ended linear member, i.e., it fixes displacement but not rotation. Thus the command implies $u = v = w = M_{nn} = 0.$ over the boundary portion $node_1$ to $node_2$.

IN-PLANE ROLLER 'boundary name' $node_1$ TO $node_2$

This support allows complete freedom of motion in the plane of the plate but restricts displacement perpendicular to the plate. Thus the command implies $N_{nx} = N_{ny} = w = M_{nn} = 0.$ over the boundary portion $node_1$ to $node_2$.

NORMAL ROLLER 'boundary name' $node_1$ TO $node_2$

This support allows complete freedom of motion perpendicular to the plate but restricts displacement in the plane of the plate. Thus the command implies $u = v = Q_{ne} = M_{nn} = 0.$ over the boundary portion $node_1$ to $node_2$.

CLAMPED EDGE 'boundary name' $node_1$ TO $node_2$

This support restricts all displacement and rotation of the plate edge. Thus the command implies $u = v = w = w,_n = 0.$ over the boundary portion $node_1$ to $node_2$.

FREE EDGE 'boundary name' $node_1$ to $node_2$

This command specifies a totally unrestrained edge, not acted upon by any external forces or moments, thus implying $N_{nx} = N_{ny} = Q_{ne} = M_{nn} = 0.$ over the boundary portion $node_1$ to $node_2$.

7. Dislocations

In multiply-connected plates, dislocations in the plane of the plate may be specified by the following command:

DISLOCATION DU δu^o DV δv^o ROTATION $\delta \omega^o$
PATH $node_1$ $node_2$ . . . . $node_n$

Symbols δu^o, δv^o, and $\delta \omega^o$ are the components of rigid body motion used to bring the positive face of the dislocation (see section 3.1-7) into coincidence with the negative face, referred to the global coordinate reference frame. The path of the dislocation extends from $node_1$, located on an exterior boundary, to $node_n$, located on an interior boundary. Each node along the dislocation must be connected to the previous one by an element, i.e., the dislocation line must always coincide with element edges.

8. Loading Specification

* LOADING 'identification' 'title'

9. Nodal loads

* NODE list LOADS FORCE X F_x Y F_y

or

* NODE LOADS
list FORCE X F_x Y F_y

.
.
.

The nodal load components F_x and F_y are the only ones applicable to this class of problems since loads perpendicular to the plate, causing plate bending, are specified indirectly through the Particular Solution commands described below.

10. Line Loads

The line load tabular command specifies a line load in the plane of the plate along a path coinciding with element edges and with force intensity components varying linearly between nodes along the line. Thus the force intensity can be completely specified by its values at the nodes along the line. The form of this command is as follows:

LINE LOAD

$node_1$ FORCE X f_x Y f_y

$node_2$ FORCE X f_x Y f_y

.

.

.

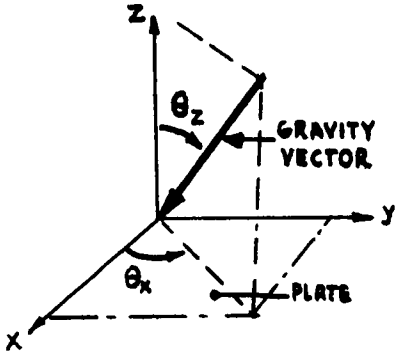
$node_n$ FORCE X f_x Y f_y

The nodes must be specified in order along the line. Data labels may be omitted if the data is given in the order shown above.

11. Gravity (or Dead) Load

The load due to the weight of the plate material (dead load) can be specified simply by specifying the orientation of the gravity vector with respect to the global axes. This specification is meaningless for a pure bending problem, however, because the component of gravity perpendicular to the plate must be specified indirectly through one of the Particular Solution commands described below. The form of the dead load specification is as follows:

GRAVITY LOAD ANGLE-X θ_x ANGLE-Z θ_z



Symbols θ_x and θ_z give the orientation of the gravity vector with respect to the x and z global axes, as shown at the left. The dead load on each element is then determined from the previously specified element thickness and density.

12. Element Loads

* ELEMENT list LOADS component type values

* ELEMENT list TEMPERATURE LOADS component value

or

* ELEMENT LOADS

.

.

.

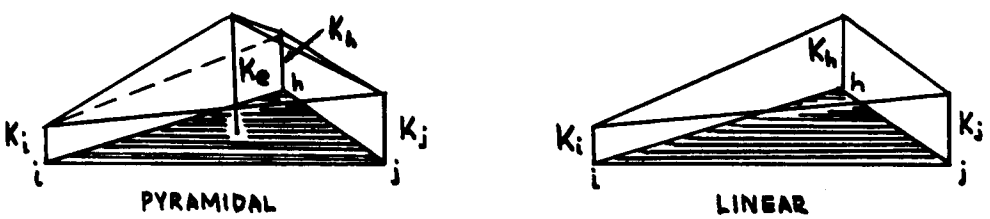
.

* ELEMENT TEMPERATURE LOADS
list component value
.
.
.

The type can be either LINEAR or UNIFORM and the applicable components are FORCE X and FORCE Y for the element loads.

13. Particular Solution for the Bending Problem

In the dual finite element stress function method for plate bending, loads perpendicular to the plate (which contribute to the bending of the plate) are specified indirectly through the two functions K_x and K_y (see page 20 or reference 7). Except in the case of a uniform load over a homogeneous plate, the nodal and optionally also the element-centered values of K_x and K_y must be explicitly specified by the user. These values are used to obtain an approximate integration of K_x and K_y over the area of each element. If the element-centered value is prescribed, a pyramidal approximation is used for the integration. Otherwise, a linear approximation is used.



Since the particular solution is the equivalent of a load for the bending problem, it applies to the most recent LOADING specification given. Thus a number of different particular solutions may be specified for the bending problem, just as a number of different loading conditions may be specified for the stretching problem. The form of the particular solution specification is as follows:

BENDING PARTICULAR SOLUTION
NODES list KX K_x KY K_y
ELEMENTS list KX K_x KY K_y
.
.
.

.
.
.

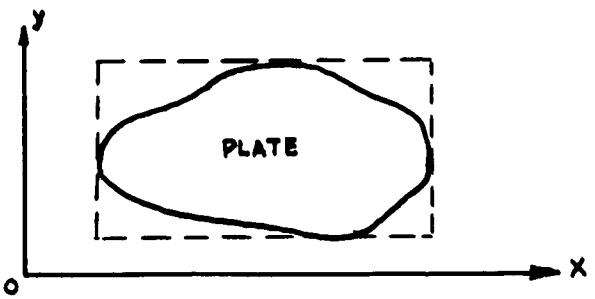
END

The list may be a single entry or a string of up to ten entries, separated by commas or blanks. The word END must appear on the line following the last tabular entry.

In the case of a uniform load over a homogenous plate, a standard particular solution may be specified by use of the following command:

PARTICULAR SOLUTION UNIFORM LOAD p_z C c

Symbol p_z is the value of the uniform load in the -z direction. Symbol c is the proportion of the load taken by a family of strips in the x-direction (see page 20 or reference 7 for a more detailed explanation). In the determination of the standard particular solution, the system "fits" the plate shape into a rectangle whose edges are parallel to the global x and y axes and are simply supported. Thus the proportionality constant ($0. \leq c \leq 1.$) should reflect somewhat the plate's proportions in the x and y directions as well as the known qualitative behavior of the plate in bending.



14. Loading List Specification
    * LOADING LIST 'loading$_1$' 'loading$_2$' . . . .
    * LOADING LIST ALL
    * LOADING LIST ALL BUT 'loading$_1$' 'loading$_2$' . . .

15. Analysis Command
    * FINITE ANALYSIS

16. Output Command
    * LIST STRESSES, STRAINS, PRINCIPAL STRESSES, PRINCIPAL STRAINS, NODAL DISPLACEMENTS, ALL

CHAPTER 5

Conclusions and Recommendations

The dual finite element stress function method discussed in this work has been shown by Elias ((Ref. 7) to be equal to, or better than, the displacement method for approximating the exact answers to some plate bending problems. The dual stress function method involves only two equations per node whereas the bending displacement method involves three equations. This fact alone makes the stress function method desireable from a computer time consumption viewpoint. When fully operational, the computer system developed in this thesis should serve as a good tool for continuing the study of the stress function method and its comparison to the displacement methods.

There are, however, certain characteristics of some of the bending problem boundary conditions that may detract from the effectiveness of the dual stress function method. In the stretching problem, all but one of the boundary conditions discussed result in only additions to the existing non-zero elements of the global structural stiffness matrix, thus maintaining the symmetry. Only strain boundary conditions result in certain rows being replaced without changing the corresponding columns, thus destroying the symmetry of the global stiffness matrix and requiring the entire matrix to be stored in the computer instead of just the lower half. This change in storage in turn requires a different solver routine to be used. In the bending problem, stress boundary conditions are dual of strain boundary conditions in the stretching problem and thus result in a non-symmetric global flexibility matrix. Since both types of mixed boundary conditions for the bending problem also involve part of the stress boundary conditions formulation, they, too, result in non-symmetric global matrices. In addition, the support stress functions U^S and V^S of the edge beam boundary conditions for the bending problem are not known quantities specified by the user, as are their dual quantities u^S and v^S of elastic boundary conditions of the stretching problem. Instead, they must be determined by a somewhat complicated procedure (pages 52-55) that also results in a non-symmetric global matrix. The possible adverse effects of the non-symmetric storage and solver requirements of some of

the bending problem boundary conditions should be investigated.

The flexibility and modularity of the system developed in this work will allow it to be modified and extended as the need requires. Routines for specifying standard plate shapes, sizes, and finite element discretizations could be added easily, as well as more types and options of boundary conditions, more types of standard particular bending solutions, and more sophisticated complete output routines. At the time of this writing, the computer system developed in this thesis is not completely debugged and operational.

REFERENCES

1. Ferrante, A. J., "A System for Finite Analysis," S. M. Thesis, Civil Engineering Department, MIT, January 1967.

2. Clough, R. W., "The Finite Element Method in Structural Mechanics—Philosophy of the Finite Element Procedure," Stress Analysis, Chapter 7, edited by O. C. Zienkiewicz and G. G. Holister, John Wiley and Son, 1965.

3. Connor, J. J., "Lecture Notes on the Finite Element Displacement Method," MIT, 1966 (unpublished).

4. Lundberg, R. C., "A Finite Element Analysis of Plates and Shells," S. M. Thesis, Civil Engineering Department, MIT, August 1966.

5. Elias, Z. M., "On the Duality Between the Problems of Stretching and of Bending of Plates," Report R66-43, Department of Civil Engineering, MIT, September 1966.

6. Zienkiewicz, O. C., "Finite Element Procedures in the Solution of Plate and Shell Problems," op. cit. Ref. 2, Chapter 8.

7. Elias, Z. M., "Dual Finite Element Method for Stretching and Bending of Plates," Report R67-16, Department of Civil Engineering, MIT, May 1967.

8. "ICES: Concepts and Facilities," Department of Civil Engineering, MIT, 1965.

9. "ICES: Programmer's Guide," Department of Civil Engineering, MIT, 1965.

10. McCracken, D. D., "A Guide to FORTRAN IV Programming," John Wiley and Sons, Inc., 1965.

11. Timoshenko, S., and Woinowsky-Krieger, S., "Theory of Plates and Shells," McGraw Hill Book Co., Inc., 1959.

12. Logcher, R., "ICES-STRUDL I-Logic Reference Manual," Department of Civil Engineering, MIT, (unpublished).

APPENDICES

APPENDIX 1

Additions to the Finite Element Analyzer Data Structure

Appendices 1, 2, 3, and 7 are intended to serve as a detailed description of the internal characteristics of the programming capabilities developed in this thesis. Since these capabilities are a subset of the general Finite Element Analyzer programming system, the detailed documentation of the Finite Element Analyzer (Reference [1], pages 53-105) should be understood before proceeding to the documentation contained herein. For ease of cross-reference, the numbering of appendicies in this thesis corresponds to that of Reference [1].

A. 'BDID' - One level double word array to store the alphanumeric identification of a boundary (8-character boundary name). The integer IBCØN is the length of this array. The internal number of a boundary is its position in the 'BDID' array.

Dynamic array BDID
Length = IBCØN
DEFINE BDID, 5, BOUBLE, STEP = 1
Defined in Subroutine BDINIT

B. 'BDCOND' - Three level full word array of boundary conditions referenced by boundaries, nodes in order around the boundary, and data.

BDCOND(I,J,1) = internal node number (input phase)
BDCOND(I,J,2) = type of boundary condition for the stretching problem

= 1 displacement
= 2 stress
= 3 elastic boundary
= 4 edge beam
= 5 mixed
= 6 rigid-fixed
= 7 rigid-pinned
= 8 rigid-free

BDCOND(I,J,3) = type of boundary condition for the bending problem

= 1 displacement

= 2 stress

= 3 elastic

= 4 edge beam

= 5 mixed/1

= 6 mixed/2

BDCOND(I,J,4) = specified values of the boundary

BDCOND(I,J,5) conditions, located as shown in

⋮ the table below.

Storage location	Boundary Condition Type							
	1	2	3	4	5	6	7	8
(I,J,4)	u_j	N_{nx}	u^s	E	N_{nr}	u_1	u_p	F_{xp}
(I,J,5)	v_j	N_{ny}	v^s	I_z	u_r	v_1	v_p	F_{yp}
(I,J,6)			k_{xx}	N_x	θ_r	ω_1	x_p	x_p
(I,J,7)			k_{xy}	N_y			y_p	y_p
(I,J,8)			k_{yx}				M_p	M_p
(I,J,9)			k_{yy}			χ_{zs}	χ_{zs}	χ_{zs}
(I,J,10)						ε_{ss}	ε_{ss}	ε_{ss}
(I,J,11)	w_j	Q_{ne}	w^s	Q_{ne}	w_j	Q_{ne}		
(I,J,12)	$w_{,n}$	M_{nn}	$w^s_{,n}$	M_{nn}	M_{nn}	$w_{,n}$		
(I,J,13)			K_{zz}	I_n				
(I,J,14)			K_{ss}	J				
(I,J,15)				G				

If the problem is only bending (IPROB = -1) locations (I,J,11) through (I,J,15) are shifted up to (I,J,4) through (I,J,8).

Dynamic Array BDCØND

DEFINE BDCOND, 5, PØINTER, STEP = 1

DEFINE BDCOND(I), 10, POINTER, STEP = 10

DEFINE BDCOND(I,J), 5, FULL, STEP =5

Defined in Subroutines BDINIT and NBDASS

C. 'DISLØC' - Two level full word array of dislocations, appearing in the order specified, i.e., DISLØC(3,N) refers to the third dislocation specified by the user. The data is stored as follows:

DISLØC(I,1) = δu^o

(I,2) = δv^o

(I,3) = $\delta \omega^o$

(I,4) = position in JTID of first node on the dislocation

(I,j+3) = jth node on the dislocation

(I,n+3) = nth (last) node on the dislocation

Dynamic array DISLØC

DEFINE DISLØC, 5,POINTER, STEP = 1

DEFINE DISLØC(I), 10, FULL, STEP = 10

Defined in Subroutine DISLCP

D. 'PBSØLN' - Three level full word array of nodal values of the particular bending solution functions K_x and K_y. Referenced by loadings, nodes, and data.

PBSØLN(I,L,1) = K_x

PBSØLN(I,L,2) =Ky

Dynamic array PBSØLN

Length = LEXTN

Length = 2

DEFINE PBSOLN, LEXTN, POINTER, STEP = 5

DEFINE PBSOLN(LEXTN), JEXTN, POINTER

DEFINE PBSOLN(LEXTN,I), 2, FULL

Defined in Subroutine PARTIC or STDPSL

E. 'PBSØLE' - Three level full word array of element-centered values of K_x and K_y (if IPRTIC(LEXTN)=0) or of exact integrals of K_x and K_y over the surface area of the element (if IPRTIC(LEXTN)=1). Referenced by loadings, elements, and data.

PBSØLE(L,I,1) $=K_x$ or $\int K_x dA_i$

PBSØLE(L,I,2) $=K_y$ or $\int K_y dA_i$

Dynamic array PBSØLE

Length = LEXTN

Length = JEXTN

Length = 2

DEFINE PBSOLE, LEXTN, POINTER, STEP =5

DEFINE PBSOLE(LEXTN), JEXTN, POINTER

DEFINE PBSOLE(LEXTN,I), 2, FULL

Defined in Subroutine PARTIC or STDPSL

F. 'IPRTIC" - One level half word array indicator of the type of particular bending solution specified by the user ofr each loading condition.

IPRTIC(L)	= 0	nodal and element-centered values specified by the user explicitly.
IPRTIC(L)	= 1	standard solution for uniform load and homogeneous plate requested by the user.

Dynamic array IPRTIC

Length = LEXTN

DEFINE IPRTIC, 5,HALF, STEP = 5

Defined in Subroutine PARTIC or STDPSL

G. 'SFTEMP' - Three level full word array of the nodal values of the stress functions for each loading. Referenced by loadings, nodes, and data.

SFTEMP(L,I,1) = U_i

SFTEMP(L,I.2) = V_i

Dynamic array SFTEMP

Length = LEXTN

Length = JEXTN

Length = 2

DEFINE SFTEMP, LEXTN, JEXTN, 2

Defined in Subroutine STNBKS

H. 'RNDTEM' - Three level full word array for temporary storage of nodal displacements during the bending solution phase of a general plate problem. Referenced by loadings, nodes, data.

RNDTEM(L,I,1) $=u_i$

RNDTEM(L,I,2) $=v_i$

Dynamic array RNDTEM

Length = LEXTN

Length = JEXTN

Length = 2

DEFINE RNDTEM, LEXTN, JEXTN, 2

Defined in Subroutine STNBKS

I. Scalars in CØMMØN:

1. IPROB = indicator of problem type and solution phase
   - = 0 stretching problem
   - = 1 bending problem
   - = 2 general plate problem (both stretching and bending)
   - = + stretching solution phase
   - = - bending solution phase

2. IBCON = connectivity of plate (number of closed boundary curves

3. NSYM = indicator of type of stiffness matrix to be used in solution
   - = 1 symmetric
   - = 2 non-symmetric

APPENDIX 2

Revised and Extended CØMMØN Map

This appendix presents the complete map of CØMMØN storage area used by the dual finite element plate analysis subroutines. The map is an extension of the Finite Element Analyzer CØMMØN map found in Appendix 2 of reference [1].

Name	Rel. Add.	Displacement Hex.	Displacement Dec.	Remarks
	ICES CØMMØN POOL			
QQDUB(2)	1-2	000	0000	
ICOM	3	008	0008	
IERROR	4	00C	0012	
ICOML	5	010	0016	
QQCOM(75)	6-80	014	0020	
	SCRATCH CØMMØN POOL			
I1 to I36	81-116	140	0320	CDL Scratch Common Pool (D1 to D10 are double words)
T1 to T36	117-152	1DO	0464	
D1 to D10	153-172	260	0608	
TEMP1(9)	173-180	2B0	0688	
NSOL	181	2D0	0720	Number of nodes at which displacements are not fully prescribed
TEMP2(48)	182-230	2D4	0724	
LEXT	231	398	0920	Independent active loadings
(pointer)	232	39C	0924	
POINT1(4)	233-236	3A0	0928	
IBAND	237	3B0	0944	Semibandwidth for hyper columns of stiffness matrix
(pointer)	238	3B4	0948	
IFDT	239	3B8	0952	bit picture of stiffness matrix
(pointer)	240	3BC	0956	

Name	Rel. Add.	Displacement hex.	Displacement Dec.	Remarks
KDIAG	241	3C0	0960	Diagonal submatrices of stiff. matrix
(pointer)	242	3C4	0964	
KOFDG	243	3C8	0968	Off-diagonal submatrices of stiff. matrix
(pointer)	244	3CC	0972	
IOFDG	245	3D0	0976	Non-zero submatrices in each row of stiff. mat.
(pointer)	246	3D4	0980	
KPPRI	247	3D8	0984	Load vector and result vector
(pointer)	248	3DC	0988	
FCMAT	249	3E0	0992	Non-symmetric stiffness matrix elements
(pointer)	250	3E4	0996	
ICUREL	251	3E8	1000	List of non-zero rows in each hyper column of non-sym. stiff. mat.
(pointer)	252	3EC	1004	
IREL1	253	3F0	1008	Indicator of non-zero columns of each row of non-sym. stiff. matrix
(pointer)	254	3F4	1012	
POINT2(2)	255-256	3F8	1016	
	NON-DICTIONARY CØMMØN POOL			
FILL1(6)	257-262	400	1024	Dummy
ISCAN	263	418	1048	Scanning mode indicator
FILL2(52)	264-315	41C	1052	Dummy
CFLEN	316	4EC	1260	Conversion factor for length
CFWT	317	4F0	1264	Conv. factor for weight
CFANG	318	4F4	1268	Conv. factor for angle
CFTEMP	319	4F8	1272	Conv. factor for Temp.
CFTIME	320	4FC	1276	Conv. factor for time
FILL3(10)	321-330	500	1280	Dummy
	DICTIONARY CØMMØN POOL			
LDID	331	528	1320	Loading names
(pointer)	332	52C	1324	
DUM1	333	530	1328	Dummy
LEXTN	334	534	1332	Total number of loadings
LTYP	335	538	1336	Loading type
(pointer)	336	53C	1340	

Name	Rel. Add.	Displacement		Remarks
		Hex.	Dec.	
LDLIST	337	540	1344	Loading list
(pointer)	338	544	1348	
	339	548	1352	
LDTLE	340	54C	1356	Loading titles
(pointer)	341	550	1360	
JTID	342	554	1364	Node names
(pointer)	343	558	1368	
	344	55C	1372	
JEXTN	345	560	1376	Total number of nodes
JTYP	346	564	1380	Node type
(pointer)	347	568	1384	
	348	56C	1388	
DUM2(2)	349-350	570	1392	Dummy
JTXYZ	351	578	1400	Node coordinates
(pointer)	352	57C	1404	
JTLOD	353	580	1408	Nodal loads
(pointer)	354	584	1412	
FILL4(43)	355-399	588	1416	Dummy
NJ	400	63C	1596	Number of active nodes
DUM3	401	640	1600	Dummy
NLDSI	402	644	1604	Number of independent loading conditions
JF	403	648	1608	Number of degrees of freedom
ID	404	64C	1612	Problem type
FILL5(5)	405-409	650	1616	Dummy
JINT	410	664	1636	Input to analysis node correspondence
(pointer)	413	670	1648	
FILL6(26)	414-439	674	1652	Dummy
ELID	440	6DC	1756	Element names
(pointer)	441	6E0	1760	
DUM4(2)	442-443	6E4	1764	Dummy

Name	Rel. Add.	Displacement		Remarks
		Hex.	Dec.	
ELPROP	444	6EC	1772	Element properties
(pointer)	445	6F0	1776	
ELTOP	446	6F4	1780	Connectivity table
(pointer)	447	6F8	1784	
ELOADS	448	6FC	1788	Element loads
(pointer)	449	700	1792	
DUM5(10)	450-459	704	1796	Dummy
ELINT	460	72C	1836	Input to analysis element correspondence
(pointer)	461	730	1840	
ELEXT	462	734	1844	Analysis to input element correspondence
(pointer)	463	738	1848	
NBXTEL	464	73C	1852	Total number of elements
NBEL	465	740	1856	Number of active elements
NSYM	466	744	1860	Symmetry indicator
NGEN	467	748	1864	General plate indicator
ELSTDE	468	74C	1868	Standard Young's modulus
ELSTDG	469	750	1872	Standard shear modulus
ELSTCT	470	754	1876	Standard coefficient of thermal expansion
ELSTDS	471	758	1880	Standard density
ELSTPO	472	75C	1884	Standard Poisson's coefficient
FILL7(4)	473-476	760	1888	Dummy
ELSTMT	477	770	1904	Element stiffness
(pointer)	478	774	1908	
NODISP	479	778	1912	Nodal displacements
(pointer)	480	77C	1916	
STRAIN	481	780	1920	Element strains
(pointer)	482	784	1924	
STRESS	483	788	1928	Element stresses
(pointer)	484	78C	1932	
PRSTRN	485	790	1936	Element principal strains
(pointer)	486	794	1940	
PRSTRS	487	798	1944	Element principal stresses
(pointer)	488	79C	1948	

Name	Rel. Add.	Displacement		Remarks
		Hex.	Dec.	
FILL8(5)	489-493	7A0	1952	Dummy
IPROB	494	7B4	1972	Problem phase indicator
IBCON	495	7B8	1976	Boundary connectivity
BDID	496	7BC	1980	Boundary names
(pointer)	497	7C0	1984	
BDCOND	498	7C4	1988	Boundary conditions
(pointer)	499	7C8	1992	
DISLOC	500	7CC	1996	Dislocations
(pointer)	501	7D0	2000	
PBSOLN	502	7D4	2004	Nodal values of particular solution
(pointer)	503	7D8	2008	
PBSOLE	504	7DC	2012	Element values of particular solution
(pointer)	505	7E0	2016	
SFTEMP	506	7E4	2020	Nodal values of the stress functions
(pointer)	507	7E8	2024	
RNDTEM	508	7EC	2028	Temporary array of nodal displacements
(pointer)	509	7F0	2032	
IPRTIC	510	7F4	2036	Indicator of data in array PBSOLE
(pointer)	511	7F8	2040	

APPENDIX 3

Program Documentation

A3-1 General Organization

The general organization of the programming system is the same as that of the Finite Element Analyzer, as shown in Figures A3-1, A3-2, and A3-3 of Ref. 1., with the following load modules substituted for the original Finite Element Analyzer modules:

Module	replaces	Original Module
STINGEN		STEGEN
STNSAS		STEASS*
STNBCM		STJPRC
STNSSL		STSLVR*
STNBKS		STEBKS

(* indicates modules replaced only in the case of a non-symmetric global stiffness/flexibility matrix)

AE-2 Input Programs Documentation

a) CDL Programs:

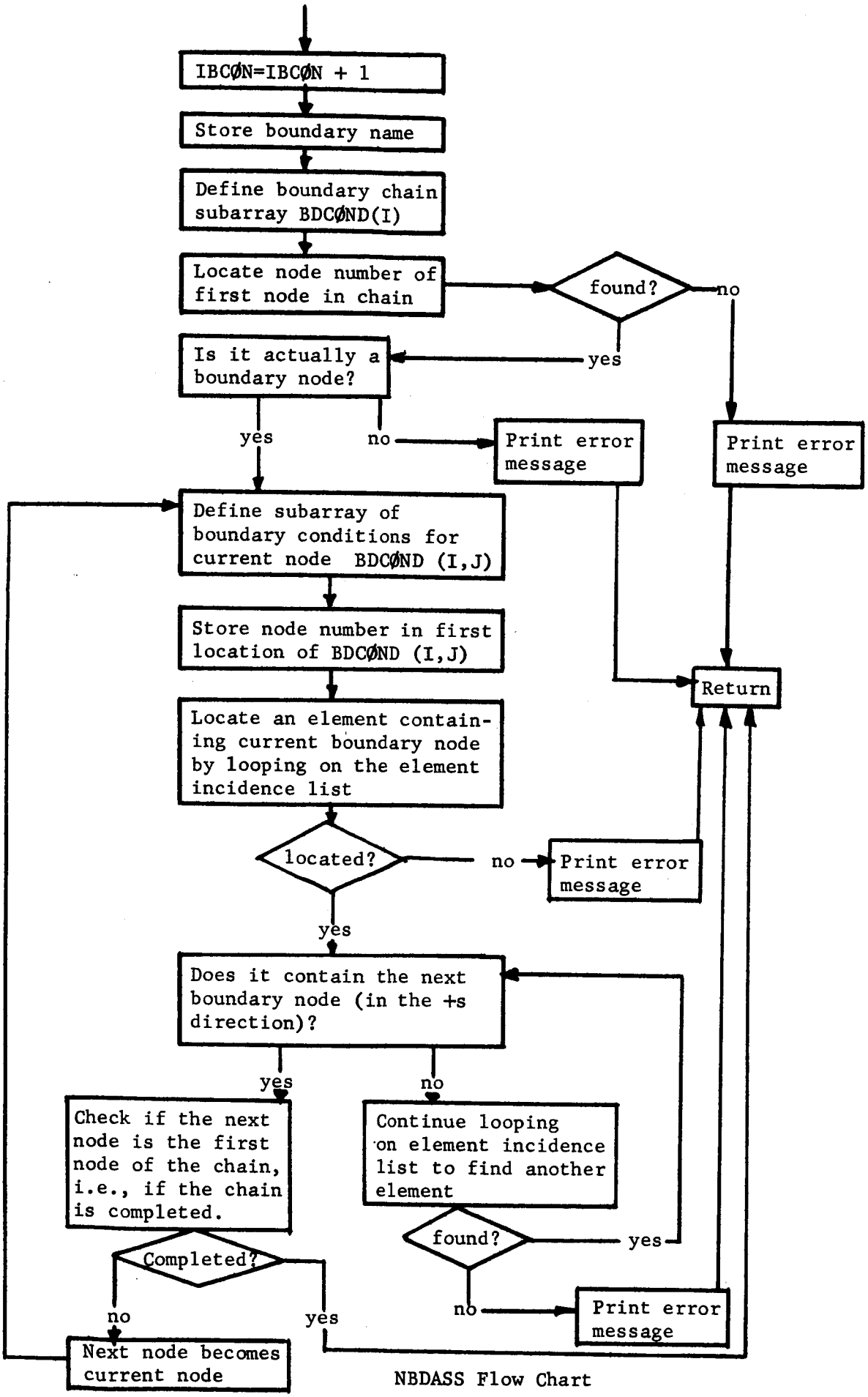
1. BOUNDARY INCIDENCES command
2. MTRAN (a CDL subroutine)
3. BOUNDARY CONDITIONS command
4. STABC (a CDL subroutine)
5. SIMPLE SUPPORT command
6. IN-PLANE ROLLER command
7. NORMAL ROLLER command
8. CLAMPED EDGE command
9. FREE EDGE command
10. LINE LOAD command
11. GRAVITY LOAD command
12. BENDING PARTICULAR SOLUTION command
13. LPROC (a CDL subroutine)
14. XROUT (a CDL subroutine)
15. DISLOCATION command

b) ICETRAN and FORTRAN input programs

Program Name:	BDINIT
Author:	D. A. Nagy
Date:	May 1967
Language:	ICETRAN
Program Description:	BDINIT processes the tabular heading command BOUNDARY INCIDENCES. It initializes the connectivity (boundary) counter IBCØN and defines the array of boundary names (BDID) and the array of boundary chains (BDCØND). It checks the common variable ID to determine what type of problem the user has specified, and sets IPROB accordingly:

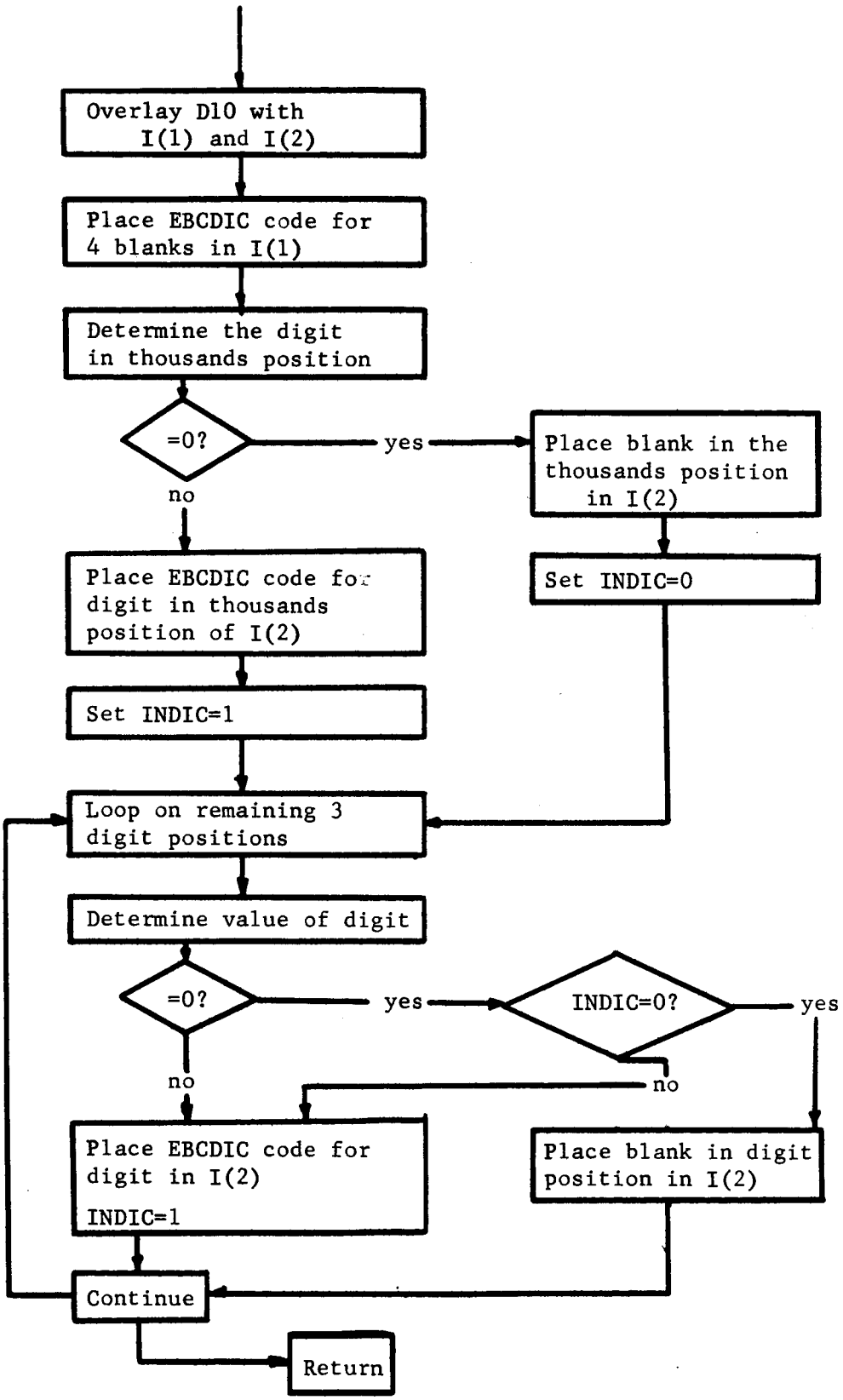
IPROB=0	plate stretching (plane stress)
IPROB=-1	plate bending
IPROB=+2	general plate (both stretching and bending)

Linkage to Program:	It is called by CDL
Usage:	STRUDL subsystem BOUNDARY INCIDENCES COMMAND
Program Output:	None
Program Length:	22 cards ICETRAN 808 bytes object program
Load Module:	BDINIT
Linkage from Programs:	None
Linkage to Programs:	None
Error Procedure:	If the user specified some TYPE other than PLANE STRESS, PLATE BENDING, or GENERAL PLATE, the following message is printed: 'BOUNDARY INCIDENCES COMMAND VALID ONLY FOR PLANE STRESS, PLATE BENDING, AND GENERAL PLATE PROBLEMS--ERROR.' ISCAN is set equal to 2.



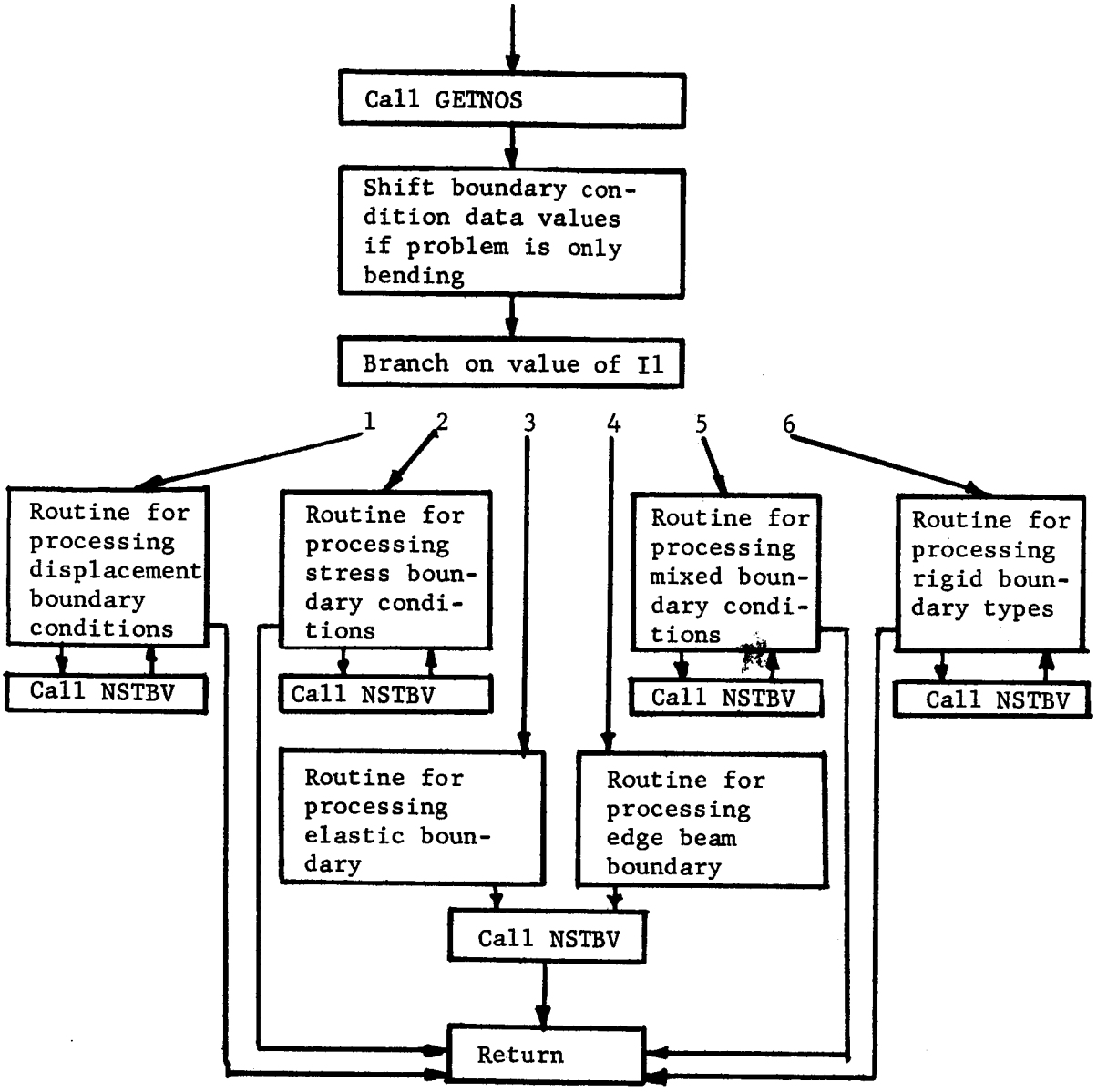
NBDASS Flow Chart

Program Name:	MTRANS
Author:	D. A. Nagy
Date:	July 1967
Language:	FØRTRAN IV
Program Description:	MTRANS translates an integer of four or less digits into alphanumeric representation.
Program Logic:	It overlays upon the double word D10 the two integers I(1) and I(2). Then it performs integer arithmetic on I(2) to create the appropriate EBCDIC (extended binary coded decimal interchange code) representation for the digits of the integer being translated.
Linkage to Program:	It is called by CDL
Usage:	STRUDL subsystem various input commands
Program Output:	None
Program Length:	26 cards FØRTRAN IV 808bytes object program
Load Module:	BDINIT, PARTIC
Linkage to Programs:	None
Linkage from Programs:	None
Error Procedure:	None



MTRANS Flow Chart

Program Name:	NBCCP
Author:	D. A. Nagy
Date:	May 1967
Language:	ICETRAN
Program Description:	NBCCP is the boundary conditions commands processor. It manages the storage of boundary condition values in their appropriate locations in the dynamic array BDCOND.
Program Logic:	It calls GETNOS to obtain the necessary information about the boundary portion being processed. It then branches to the appropriate routine for whatever type of boundary condition was specified. The branch is made on an indicator (I1) set by CDL. In each routine, it makes appropriate use of STBV to store the boundary condition values and set indicators.
Linkage to Program:	CALL NBCCP, or called by CDL
Usage:	STRUDL subsystem processing of boundary condition commands
Program Output:	None
Program Length:	224cards ICETRAN 4824bytes object program
Load Module:	BDINIT
Linkage from Programs:	NSCCP
Linkage to Programs:	GETNOS NSTBV
Error Procedure:	None



NBCCP Flow Chart

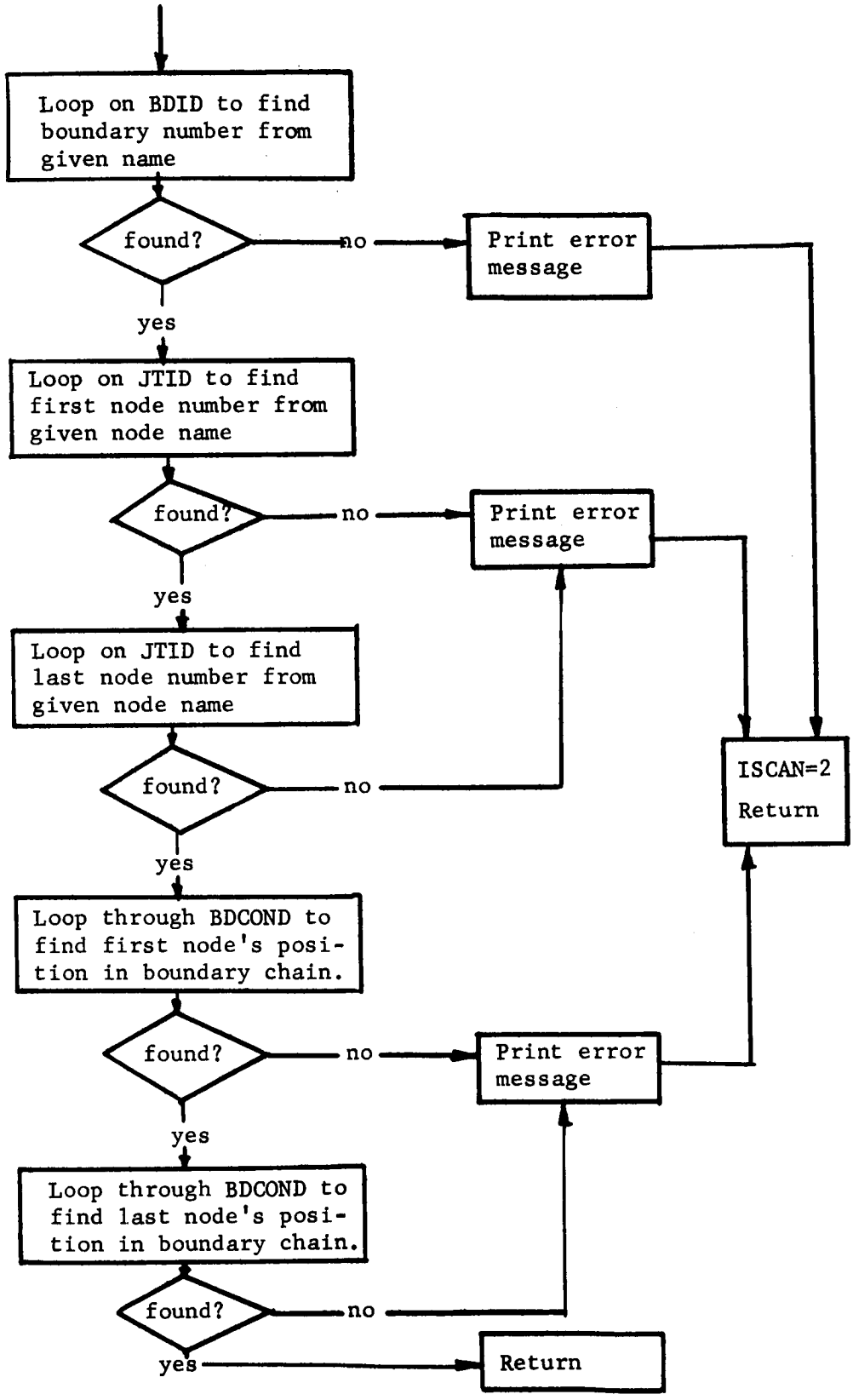
Program Name:	GETNOS
Author:	D. A. Nagy
Date:	May 1967
Language:	ICETRAN
Program Description:	Given a boundary name and the names of two nodes on that boundary, it locates the boundary number, the node numbers, and the positions of the two nodes on the boundary chain.
Program Logic:	It loops on BDID to find the boundary number (position in BDID). It loops on JTID to find the node numbers, and then it loops through the boundary chain to locate the positions of the two node numbers.
Linkage to Program:	CALL GETNOS(D1, D2, D3, I, J, K, JC, KC) where D1=the boundary name D2=first given node name D3=second given node name J =number of first given node K =number of second given node I =boundary number JC=position of first node in the given boundary chain KC=position of the second node
Usage:	STRUDL subsystem processing of boundary condition commands
Program Output:	None
Program Length:	59 cards ICETRAN 1704bytes object program
Load Module:	BDINIT
Linkage from Programs:	NBCCP
Linkage to Programs:	None
Error Procedure:	If boundary 'D1' was not previously defined, the following message is printed:

Error Procedure:
(continued)

'BOUNDARY 'D1' NOT PREVIOUSLY DEFINED--
ERROR.'

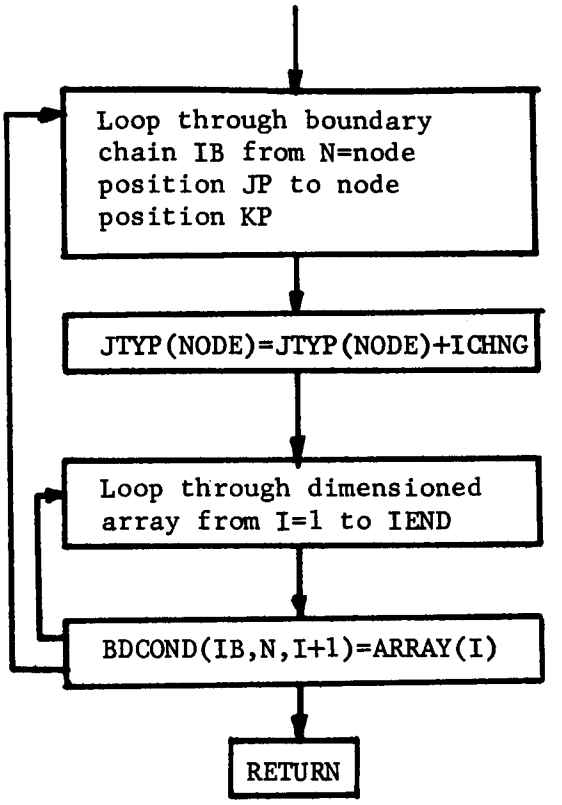
If either node 'D2' or 'D3' was not previously defined, the following message is printed:
'NODE 'name' NOT PREVIOUSLY DEFINED--
ERROR.

In all the above errors, ISCAN is set equal to 2.



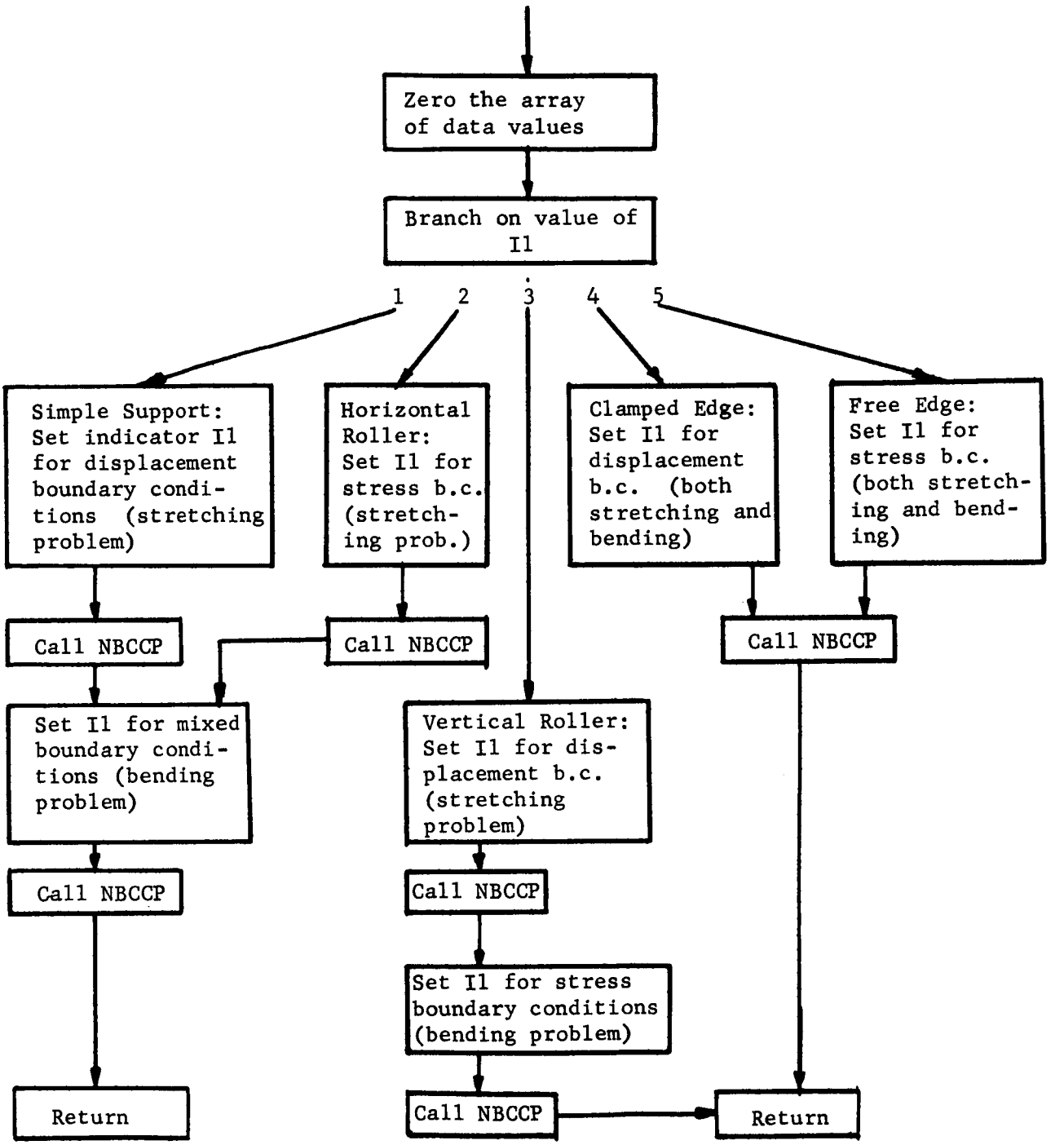
GETNOS Flow Chart

Program Name:	NSTBV
Author:	D. A. Nagy
Date:	May 1967
Language:	ICETRAN
Program Description:	NSTBV transfers boundary condition values from a dimensioned array to the dynamic array BDCOND(IB) for a string of specified nodes and sets the boundary condition indicator BDCOND(IB, NODE, 2 or 3). It then adds the integer to JTYP(NODE) necessary to indicate that boundary conditions have been specified for the node.
Program Logic:	It loops through the boundary chain nodes consecutively, performing the tasks mentioned above. At each node, it loops through the dimensioned array, transferring the values.
Linkage to Program:	CALL NSTBV(IB, JP, KP, IEND, ARRAY, ICHNG) where IB =boundary number JP =first node position KP =last node's position IEND =number of values to be transferred ARRAY =dimensioned array of values to be transferred ICHING =integer to be added to JTYP for each node
Usage:	STRUDL subsystem processing of boundary condition commands
Program Output:	None
Program Length:	20 cards ICETRAN 1000 bytes object program
Load Module:	BDINIT
Linkage from Programs:	NBCCP
Linkage to Programs:	None
Error Procedure:	None



NSTBV Flow Chart

Program Name:	NSCCP
Author:	D. A. Nagy
Date:	May 1967
Language:	FØRTRAN IV
Program Description:	NSCCP is the preliminary processor for the following commands: SIMPLE SUPPORT HORIZONTAL ROLLER VERTICAL ROLLER CLAMPED EDGE FREE EDGE
Program Logic:	It zeroes all values in the array of boundary condition values. It then branches to the appropriate routine for the command that called it (indicated by the value if I1). Each routine sets the appropriate indicator(s) and calls NBCCP to process the boundary conditions implied by the command that called NSCCP.
Linkage To Program:	It is called by CDL
Usage:	STRUDL sybsystem standard boundary type commands (see list above)
Program Output:	None
Program Length:	cards FØRTRAN IV bytes object program
Load Module:	BDINIT
Linkage from Programs:	None
Linkage to Programs:	NBCCP
Error Procedure:	None

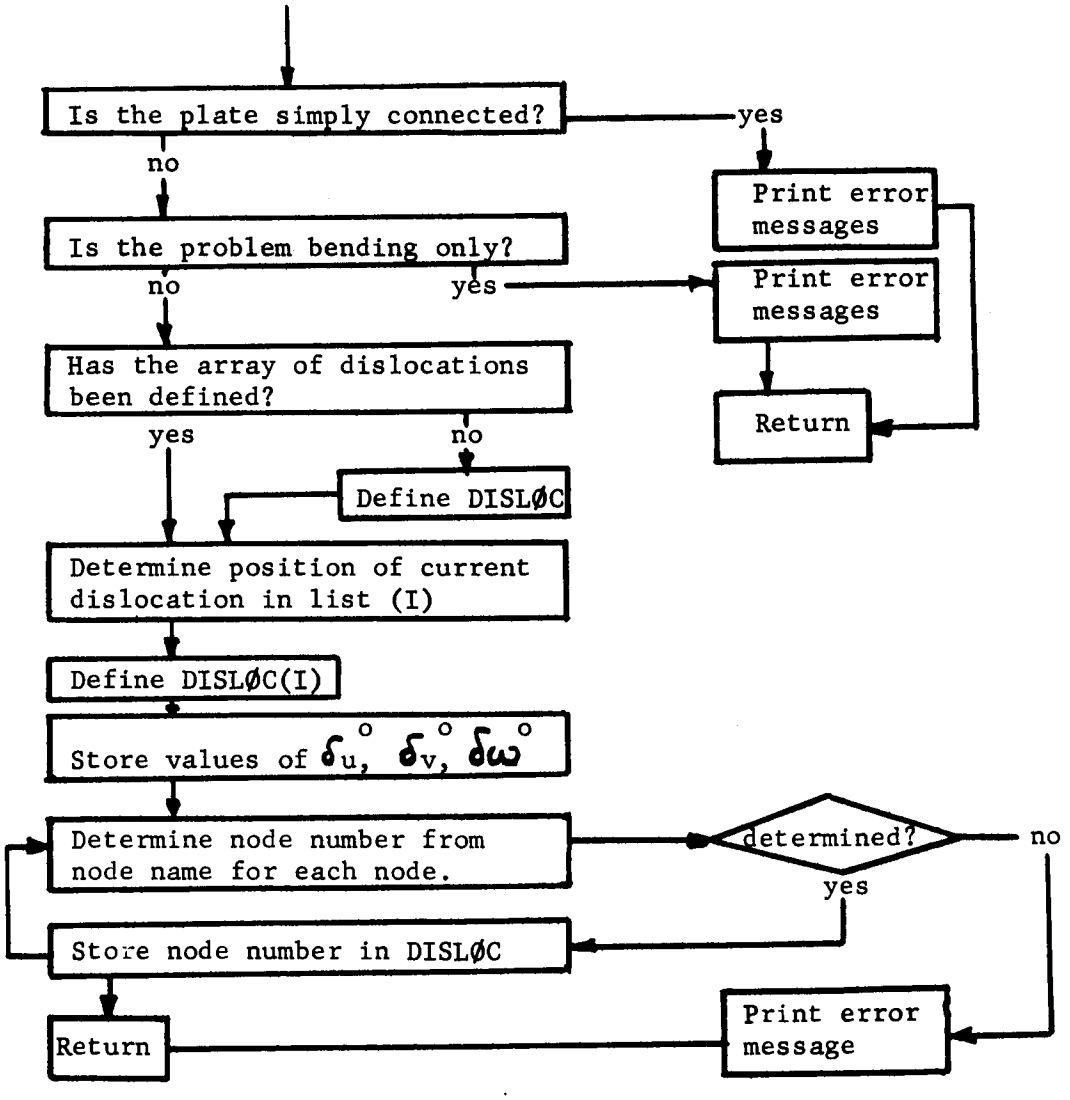


NSCCP Flow Chart

Program Name:	DISLCP
Author:	D. A. Nagy
Date:	May 1967
Language:	ICETRAN
Program Description:	DISLCP processes the DISLOCATION command.
Program Logic:	It checks if the command is appropriate, i.e., if the problem is one of plane stress in a multiply-connected plate. It checks if the pointer array of dislocations is defined, and if not, it defines it. It determines the number I of dislocations already specified, defines the subarray DISLOC(I+1) and stores the specified information in it.
Linkage to Program:	It is called by CDL
Usage:	STRUDL subsystem DISLOCATION command
Program Output:	None
Program Length:	52 cards ICETRAN 1744bytes object program
Load Module:	PARTIC
Linkage from Programs:	None
Linkage to Programs:	None
Error Procedure:	If the plate is simply-connected, the following messages are printed: 'DISLOCATION COMMAND APPLIES ONLY TO MULTIPLY-CONNECTED PLATES.' 'COMMAND WILL BE IGNORED.' If the problem is one of plate bending only, the following messages are printed: 'DISLOCATION COMMAND APPLIES ONLY TO THE PLANE STRESS PROBLEM.' 'COMMAND WILL BE IGNORED.'

Error Procedure:
(continued)

If any of the nodes on the dislocation path were not previously defined, ISCAN is set equal to 2 and the following message is printed: 'NODE 'name' WAS NOT PREVIOUSLY DEFINED--ERROR.'

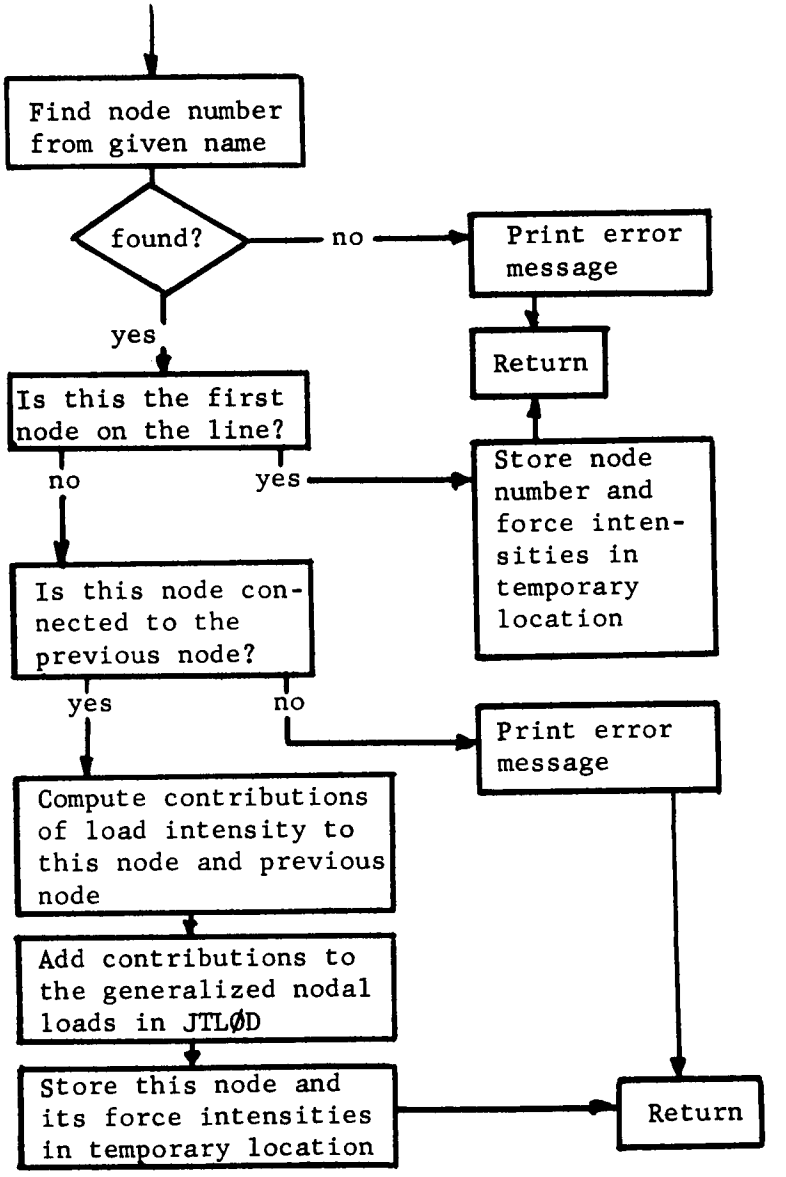


DISLCP Flow Chart

Program Name:	LINLØD
Author:	D. A. Nagy
Date:	May 1967
Language:	ICETRAN
Program Description:	LONLØD processes the LINE LØAD command, converts the specified linearly-varying line load to equivalent nodal loads, and stores in the appropriate array.
Program Logic:	It checks each node along the line to see if it has been previously defined. It checks that each node is connected to the previously-specified one by an element, i.e., that the line load coincides with element edges at all times. If these requirements are met, it computes the contribution of the line load intensity to the generalized nodal load at each node and adds it to the generalized nodal load stored in JTLØD.
Linkage to Program:	It is called by CDL
Usage:	STRUDL subsystem LINE LØAD command
Program Output:	None
Program Length:	77 cards ICETRAN 2896bytes object program
Load Module:	PARTIC
Linkage from Programs:	None
Linkage to Programs:	None
Error Procedure:	If any of the specified nodes along the line load were not previously defined, the following message is printed: 'NODE 'name' NOT PREVIOUSLY DEFINED--ERROR.' If any two nodes along the line are not connected directly by an element edge, the following message is printed: 'NODE 'name' NOT CONNECTED TO PREVIOUS

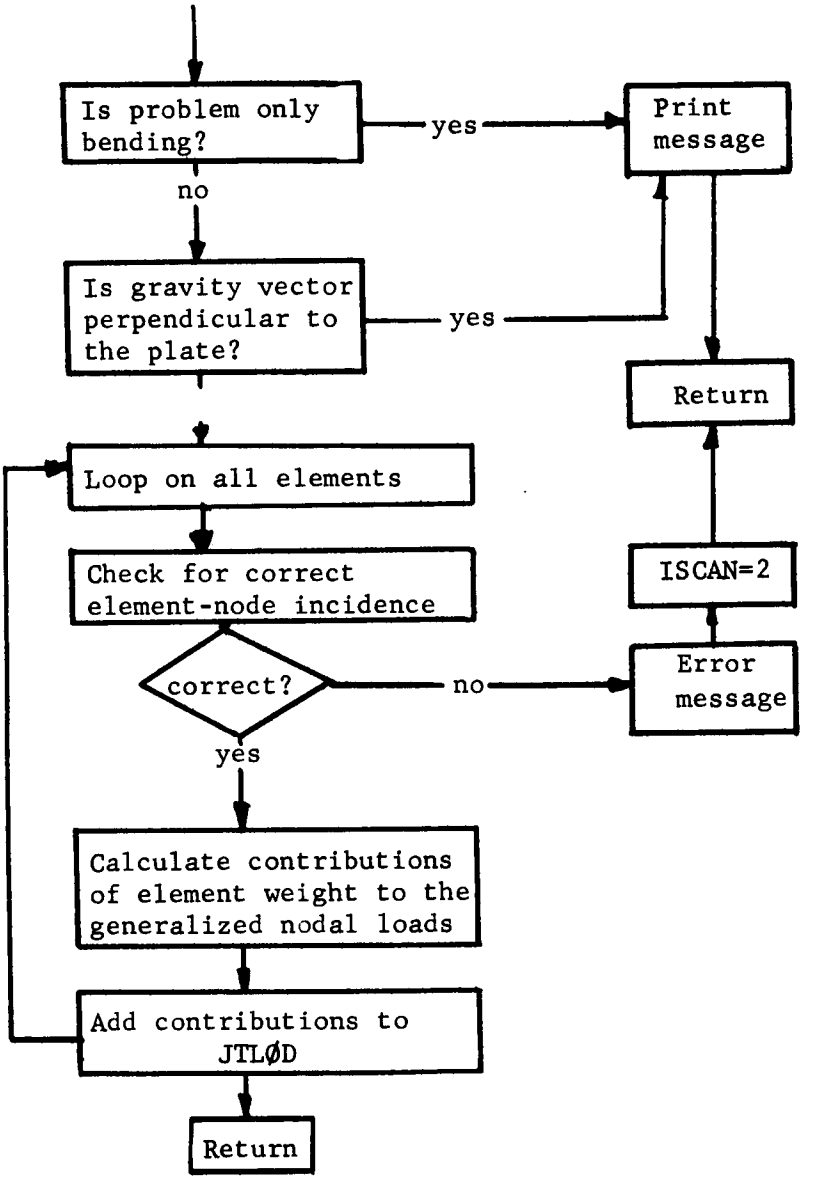
Error Procedure:
(continued)

NODE BY ANY ELEMENT--ERROR.'
In both error cases, ISCAN is set equal to 2.



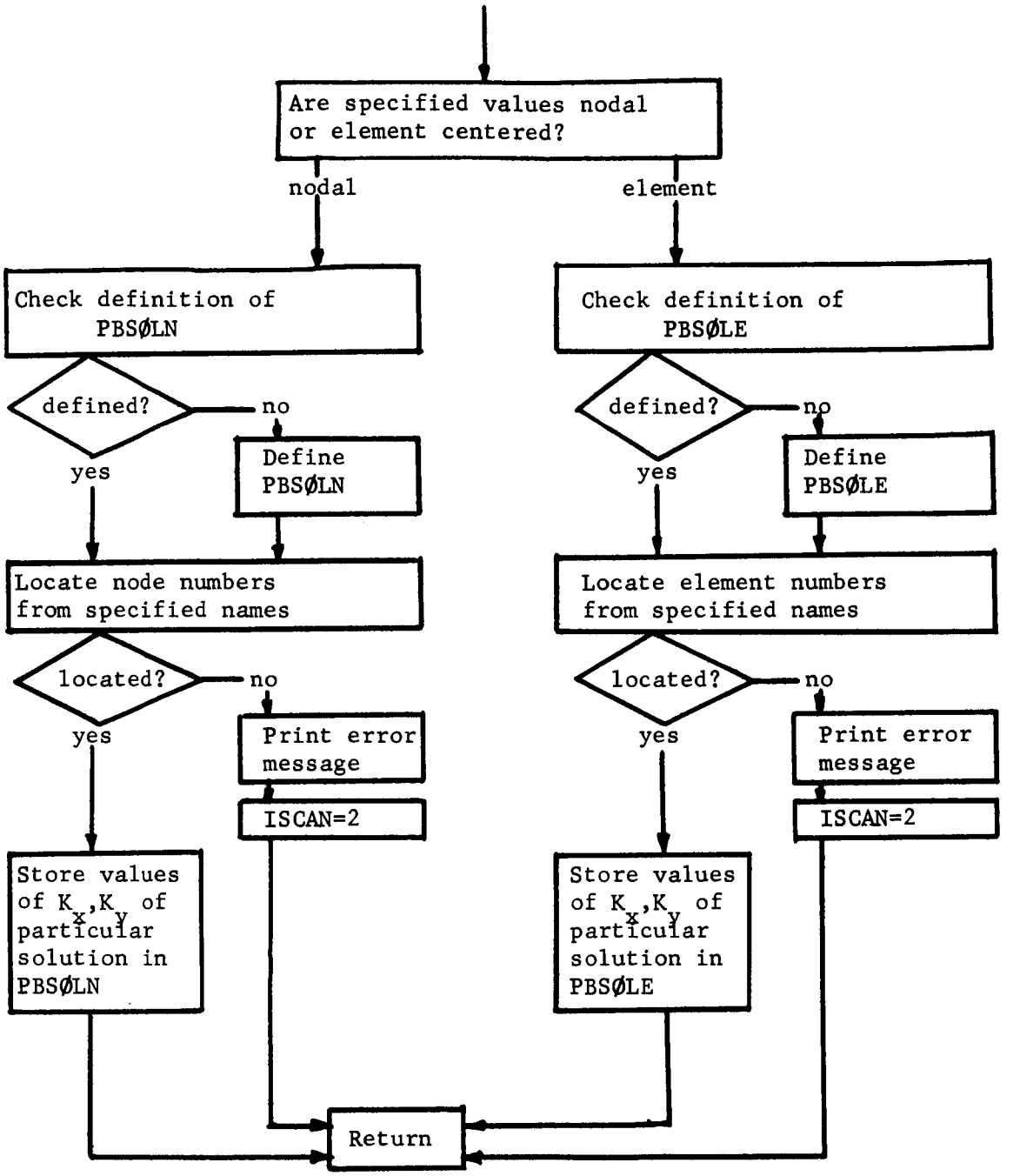
LINLØD Flow Chart

Program Name:	GRVLØD
Author:	D. A. Nagy
Date:	May 1967
Language:	ICETRAN
Program Description:	GRVLØD processes the GRAVITY LØAD command and calculates the contribution of the element weight to the generalized nodal loads at the three corners.
Program Logic:	It loops on the elements, calculating the area, volume, weight, x and y components of the gravity vector, and finally the contributions to the generalized nodal loads of the three nodes at the corners of each element.
Linkage to Program:	It is called by CDL
Usage:	STRUDL subsystem GRAVITY LØAD command
Program Output:	None
Program Length:	49 cards ICETRAN 1728bytes object program
Load Module:	PARTIC
Linkage from Programs:	None
Linkage to Programs:	None
Error Procedure:	If the problem is only one of plate bending, or if the gravity vector is perpendicular to the plate, the following messages are printed: 'COMPONENT OF THE GRAVITY LOAD PERPENDICULAR TO PLATE MUST BE SPECIFIED VIA PARTICULAR BENDING SOLUTION.' 'GRAVITY LOAD COMMAND IS IGNORED.'



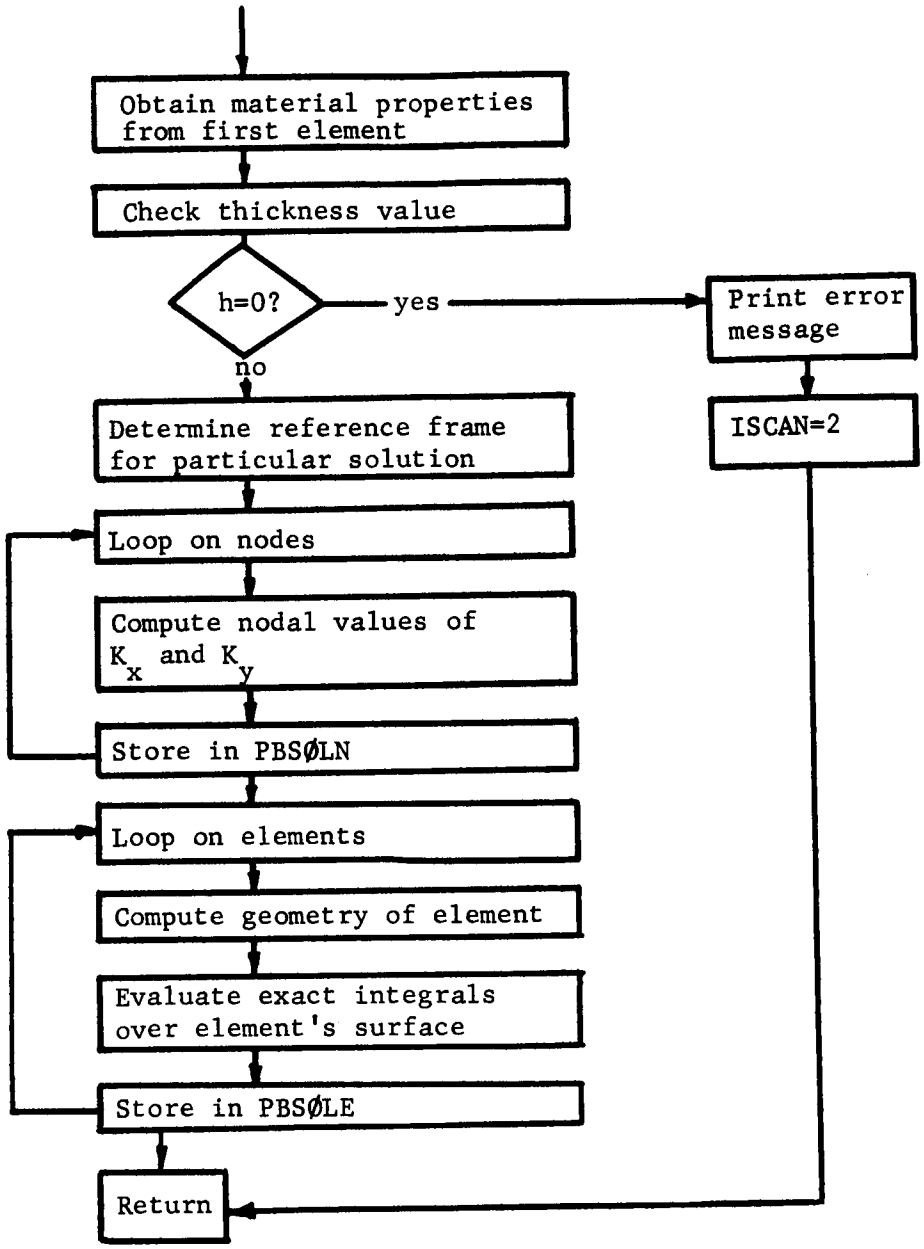
GRVLØD Flow Chart

Program Name:	PARTIC
Author:	D. A. Nagy
Date:	May 1967
Language:	ICETRAN
Program Description:	PARTIC processes the PARTICULAR BENDING SOLUTION command.
Program Logic:	It branches on the indicator I1 (set by CDL), depending on whether nodal or element-centered values of the particular bending solution are specified. It checks if the appropriate storage arrays are defined, and if not, it defines them. It then locates the node or element numbers from their given names and stores the values of the particular solution in PBSØLE (for elements).
Linkage to Program:	It is called by CDL
Usage:	STRUDL subsystem PARTICULAR BENDING SOLUTION command
Program Output:	None
Program Length:	58 cards ICETRAN 2456bytes object program
Load Module:	PARTIC
Linkage from Programs:	None
Linkage to Programs:	None
Error Procedure:	If any specified node or element was not previously defined, ISCAN is set equal to 2 and the following message is printed: 'NODE(or ELEMENT) 'name' NOT PREVIOUSLY DEFINED--ERROR.'



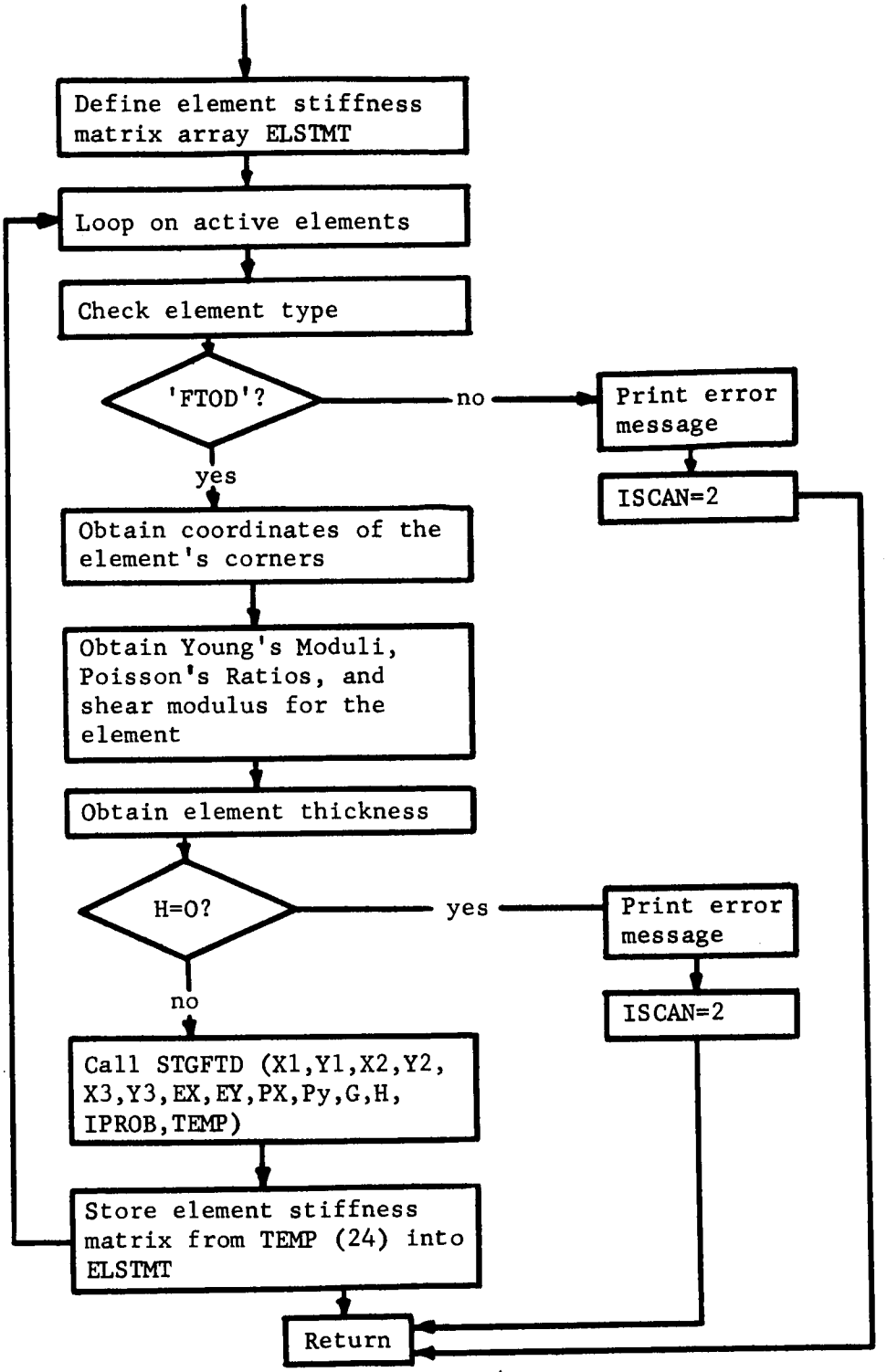
PARTIC Flow Chart

Program Name:	STDPSL
Author:	D. A. Nagy
Date:	May 1967
Language:	ICETRAN
Program Description:	STDPSL processes the PARTICULAR SOLUTION UNIFORM LOAD command. It computes the exact nodal values of the particular solution functions K_x and K_y and stores them in PBSØLE. It then evaluates the exact integral of K_x and K_y over each element and stores these values in PBSØLE. An indicator, IPRTIC, is set equal to 1 if PBSØLE contains integrals instead of element-centered values.
Program Logic:	It loops consecutively through the node list calculating the nodal values and storing them. It then loops consecutively through the element list evaluating the integrals and storing them.
Linkage to Program:	It is called by CDL
Usage:	STRUDL subsystem PARTICULAR SOLUTION UNIFORM LOAD command
Program Output:	None
Program Length:	137cards ICETRAN 5104bytes object program
Load Module:	PARTIC
Linkage from Programs:	None
Linkage to Programs:	None
Error Procedure:	Since this program only applies to plates of constant thickness, the thickness value used in computations is obtained from the first element. If this value is zero, ISCAN is set equal to 2 and the following message is printed: 'ELEMENT 'name' HAS ZERO THICKNESS--ERROR.



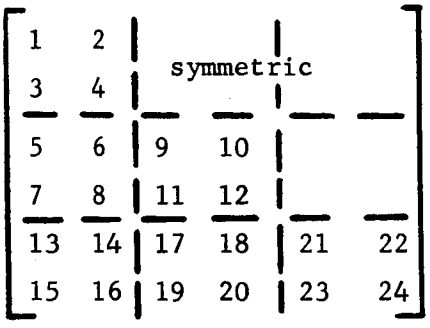
STDPSL Flow Chart

Program Name:	STNGEN
Author:	D. A. Nagy
Date:	July 1967
Language:	ICETRAN
Program Description:	STNGEN is the executive program for the element stiffness matrix generation phase.
Program Logic:	It defines the array ELSTMT in which the element stiffness matrices will be stored. It then loops on the active elements checking the element type, obtaining the necessary properties, calling STGFTD to perform the actual matrix generation for each element, and then storing the resulting matrix in ELSTMT.
Linkage to Program:	LINK TO STNGEN
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	65 cards ICETRAN 2432 bytes object program
Load Module:	STNGEN
Linkage to Programs:	STGFTD
Linkage from Programs:	STEMAI
Error Procedure:	If the element type is not 'FTØD', ISCAN is set equal to 2 and the following message is printed: 'ELEMENT'name' IS NOT OF TYPE 'FTØD'--ERROR.' If an element has zero thickness, ISCAN is set equal to 2 and the following message is printed: 'ELEMENT 'name' HAS ZERO THICKNESS--ERROR.'

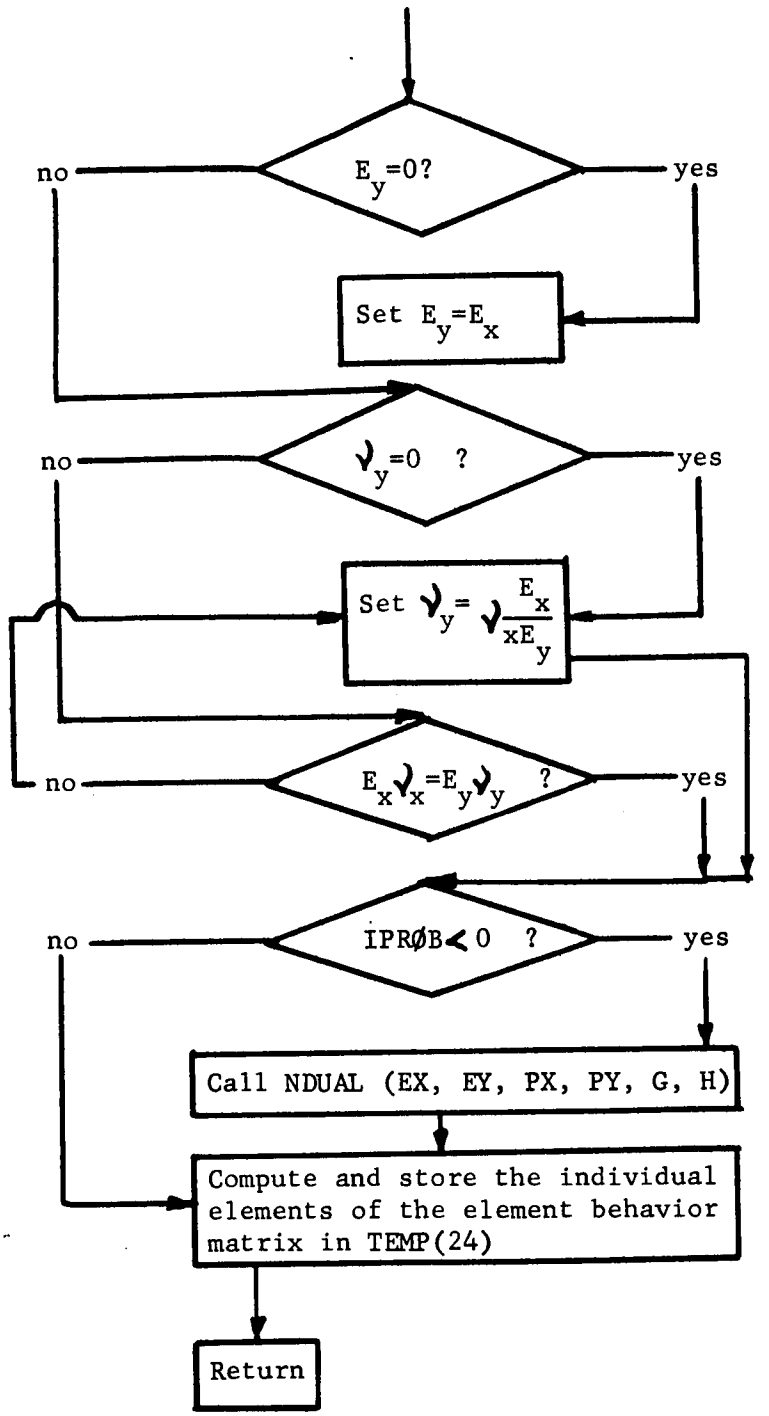


STNGEN Flow Chart

Program Name:	STGFTD
Author:	D. A. Nagy
Date:	May 1967
Language:	FØRTRAN
Program Description:	STGFTD calculates the elements of the behavior matrix for elements of the type 'FTØD' and stores them in a dimensioned array TEMP(24) according to the following correspondance:



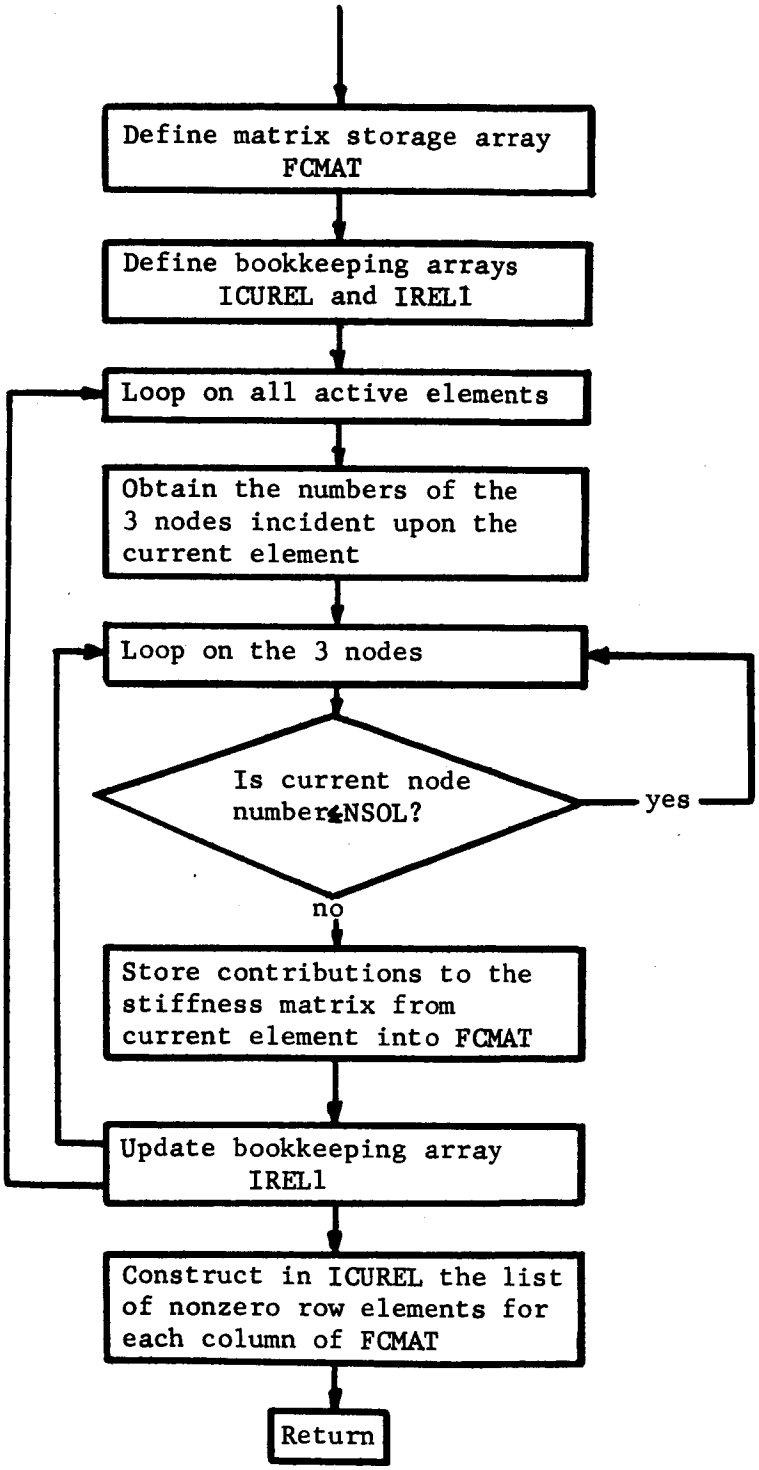
Program Logic:	STGFTD checks for consistency and adequacy of the material properties transmitted to it. If IPRØB<0, the problem is a bending problem and subroutine NDUAL is called to perform the necessary duality conversion of material properties.
Linkage to Program:	Called STGFTD
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	29 cards FØRTRAN program 1176bytes object program
Load Module:	STNGEN
Linkage to Programs:	NDUAL
Linkage from Programs:	STEGEN
Error Procedure:	None



STGFTD Flow Chart

Program Name:	NDUAL
Author:	D. A. Nagy
Date:	May 1967
Language:	FØRTRAN
Program Description:	NDUAL performs the duality conversion of material properties from the stretching problem to the bending problem (see Table I)
Linkage to Program:	CALL NDUAL (EX, EY, PX, PY, G, H)
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	9 cards FØRTRAN program 512 bytes object program
Load Module:	STNGEN
Linkage from Programs:	STGFTD
Linkage to Programs:	None
Error Procedure:	None

Program Name:	STNSAS
Author:	D. A. Nagy
Date:	August 1967
Language:	ICETRAN
Program Description:	STNSAS is the assembler of the complete global stiffness matrix for the case of non-symmetric problems.
Program Logic:	It loops on the elements, obtaining the 3 nodes incident upon each element and adding the appropriate contributions to the submatrices of the global stiffness matrix. It also constructs two bookkeeping arrays to indicate the location of non-zero submatrices in the global stiffness matrix.
Linkage to Program:	LINK TO STNSAS
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	72 cards ICETRAN 5160bytes object program
Load Module:	STNSAS
Linkage from Programs:	STEMAI
Linkage to programs:	None
Error Procedure:	None



STNSAS Flow Chart

A3-5 Solver Interface Program

Program Name:	STNSSL
Author:	D. A. Nagy
Date:	August 1967
Language:	ICETRAN
Program Description:	STNSSL is an interface program which allows the system to use a non-symmetric solver written originally for another system.
Program Logic:	The non-symmetric solver solves for only one loading condition, expects the load vector to be in a different form than KPPRI, and returns the solution values in a vector of different form from KPPRI. Thus the interface program transfers the generalized loads/rotation from KPPRI to another vector, calls the non-symmetric solver once for each independent loading condition, and transfers the solution values back into KPPRI.
Linkage to Program:	LINK TO STNSSL
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	21 cards ICETRAN 1120 bytes object program
Load Module:	STNSSL
Linkage to Programs:	STNDUM (the actual solver)
Linkage from Programs:	None
Error Procedure:	None

A3-6 Boundary Condition Programs

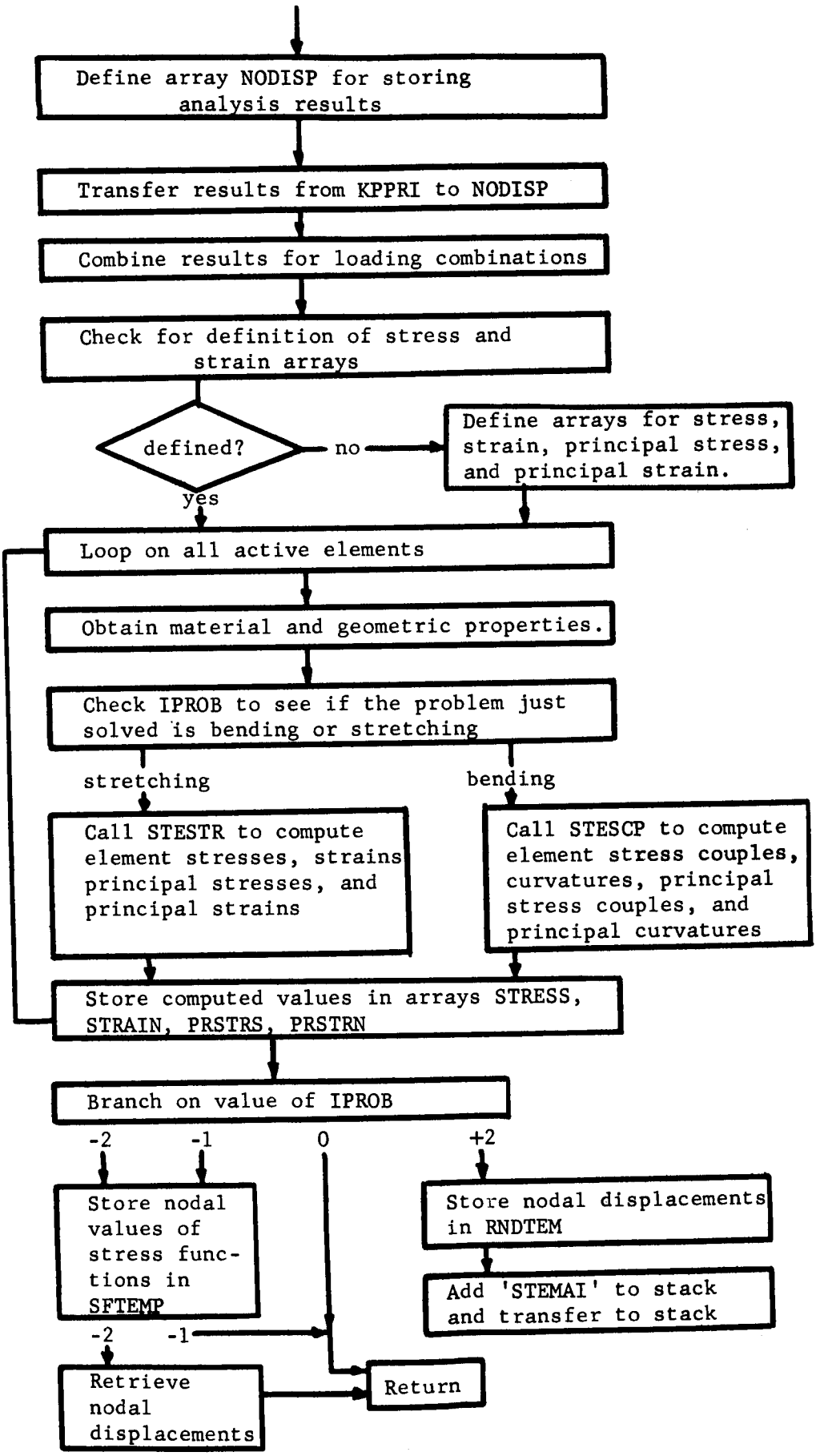
Program Name:	STNBCM
Author:	D. A. Nagy
Date:	August 1967
Language:	ICETRAN
Program Description:	STNBCM is the executive program for the boundary conditions processing phase.
Program Logic:	It determines whether the current problem is bending or stretching and then calls the appropriate subroutine to generate either the generalizee nodal loads or rotations vector. It then loops on all the boundaries, processing each different boundary condition portion. It determines the type of boundary condition and then calls a dictionary program that in turn calls the appropriate subroutine to modify the global stiffness/flexibility matrix and load/rotation vector.
Linkage to Program:	LINK TO STNBCM
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	57 cards ICETRAN 2072 bytes object program
Load Module:	STNBCM
Linkage to Programs:	STNSLV STNBLV DICTS DICTB
Linkage from Programs:	None
Error Procedure:	If the boundary conditions for some portion of a boundary are not specified, the following message is printed: 'CONDITIONS FOR BOUNDARY 'name' NOT COMPLETELY SPECIFIED--ERROR'

Program Names:	STNSLV, STNBLV
Author:	D. A. Nagy
Date:	August 1967
Language:	ICETRAN
Program Descriptions:	STNSLV constructs the generalized nodal load vector and stores it in the array KPPRI. STNBLV constructs the generalized nodal rotations vector, including the contribution of the particular solution, and stores it in KPPRI. Only the active, independent loadings are considered.
Linkage to Programs:	CALL STNSLV or CALL STNBLV
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Lengths:	STNSLV: 33 cards ICETRAN 1232 bytes object program STNBLV: 71 cards ICETRAN 1280 bytes object program
Load Module	STNBCM
Linkage from Programs:	STNBCM
Error Procedure:	None

Program Names:	DICTS, DICTB
Author:	D. A. Nagy
Date:	August 1967
Language:	FORTRAN IV
Program Descriptions:	DICTS and DICTB are dictionary programs that branch to the appropriate subroutine for processing stretching or bending boundary conditions, respectively, over a boundary portion starting at node NODE of boundary IBN. The type of condition is IBC.
Linkage to Programs:	CALL DICTS (IBN,IBC,NODE) CALL DICTB (IBN,IBC,NODE)
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	DICTS: 19 cards FORTRAN IV DICTB: 15 cards FORTRAN IV
Load Module:	STNBCM
Linkage to Programs:	DICTS to SDISPL, SSTRESS, SELAST, SEDGEB, SMIXED, SRIGID DICTB to BDISPL, BSTRESS, BELAST, BEDGEB, BMIXID, BMIX2D
Linkage from Programs:	STNBCM
Error Procedure:	None

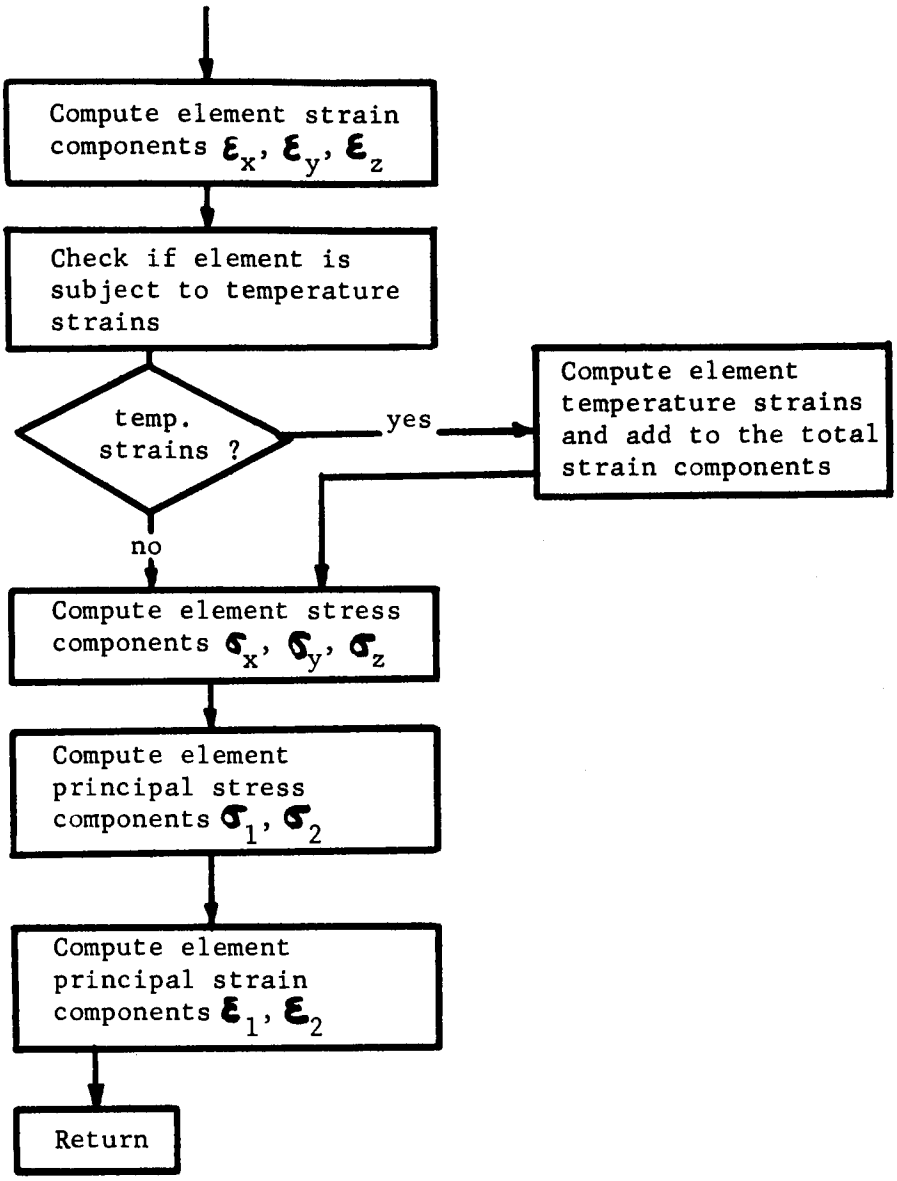
Program Names:	SDISPL,SSTRES,SELAST,SEDGEB,SMIXED,SRIGID, BDISPL,BSTRES,BELAST,BEDGEB,BMIXID,BMIX2d
Author:	D. A. Nagy
Date:	August-September 1967
Language:	ICETRAN
Program Descriptions:	These programs are the routines for processing the boundary condition portions of each boundary. The modifications to the global stiffness/flexibility matrix and loads/rotations vector are as given in Chapter 3.
Linkage to Programs:	CALL Program (IBN, NODE) where IBN= number of the boundary NODE= position of first node of the portion on the given boundary
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Load Module:	STNBCM
Linkage to Programs:	None
Linkage from Programs:	DICTS, DICTB
Error Procedure:	None

Program Name:	STNBKS
Author:	D. A. Nagy
Date:	June 1967
Language:	ICETRAN
Program Description:	STNBKS is the executive program for the backsubstitution phase.
Program Logic:	It transfers the results of the analysis from KPPRI to NØDISP and combines the results for the various loading combinations. It then loops on all active elements, obtaining the material and geometric properties necessary for the computation of element stresses and strains. It calls STESTR (for the bending problem) to compute the stresses, strains, principal stresses and strains, stress couples, and curvatures for each element. Next it either stores the nodal displacements in a temporary array or retrieves them from that array or returns, depending on the type of problem just solved. Finally, if the problem just solved was the stretching part of a general plate problem, it transfers, to STEMAI (the main executive of the Finite Element Analyzer) to begin the bending part.
Linkage to Program:	LINK TO STNBKS
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	140cards ICETRAN 6448bytes object program
Load Module:	STNBKS
Linkage to Programs:	STESTR, STESCP, STEMAI
Linkage from Programs:	STEMAI
Error Procedure:	None



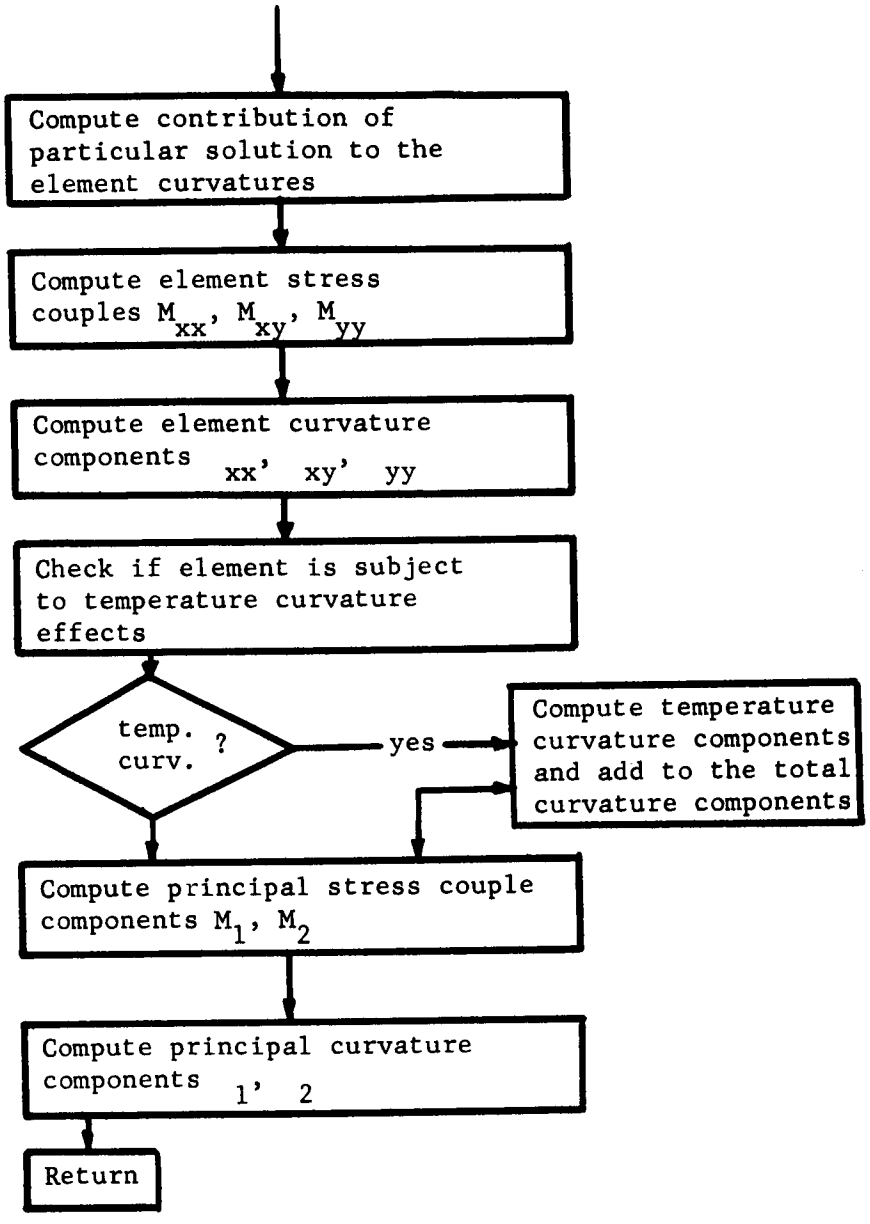
STNBKS Flow Chart

Program Name:	STESTR
Author:	D. A. Nagy
Date:	June 1967
Language:	ICETRAN
Program Description:	STESTR computes the stresses, strains, principal stresses, and principal strains for an element given the material and geometric properties of the element.
Linkage to Program:	CALL STESTR (EX,EY,PX,CX,CY,G,L,N,AR,A, B,U,V,E,S,PS,PE)
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	31 cards ICETRAN program 1408bytes object program
Load Module:	STNBKS
Linkage from Programs:	STNBKS
Linkage to Programs:	None
Error Procedure:	None



STESTR Flow Chart

Program Name:	STESCP
Author:	D. A. Nagy
Date:	June 1967
Language:	ICETRAN
Program Description:	STESCP computes the stress couples, curvatures, principal stress couples, and principal curvatures for an element given the material and geometric properties of the element. It includes the contribution of the particular solution.
Linkage to Program:	CALL STESCP (EX,EY,PX,CX,CY,G,H,L,N,AR, A,B,U,V,E,X,PS,PE)
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	None
Program Length:	58 cards ICETRAN program 2600bytes object program
Load Module:	STNBKS
Linkage from Programs:	STNBKS
Linkage to Programs:	None
Error Procedure:	None



STESCP Flow Chart

Program Name:	STNOUT
Author:	D. A. Nagy
Language:	ICETRAN
Program Description:	STNOUT is a temporary subroutine to output the nodal displacements, element stresses, stress couples, strains, curvatures, principal stresses, principal stress couples, principal strains, and principal curvatures for each loading condition of the problem just solved. It is included in the back substitution load module and is called without the request of the user. It may be replaced at a later date by a more complete and selective set of output routines called explicitly by the user.
Linkage to Program:	CALL STNOUT
Usage:	STRUDL subsystem Finite Element Analyzer
Program Output:	For each active loading condition, it prints out the nodal displacements (x and y components only), element stresses, stress couples, strains, curvatures, principal stresses, principal stress couples, principal strains, and principal curvatures.
Program Length:	88 cards ICETRAN 2280 bytes object program
Load Module:	STNBKS
Linkage to Programs:	None
Linkage from Programs:	None
Error Procedure:	None

APPENDIX 4

Table of Symbols in Alphabetical Order

a_i = x component of side i considered as a vector

A = scalar defined by Eq (218)

$\underline{\underline{A}}_k$ = matrix function of submatrices of the global flexibility matrix, defined by Eq (193)

A_n = middle surface area of triangular element n

A_k = cross-sectional area of an edge beam

b_i = y component of side i considered as a vector

$\underline{\underline{B}}_k$ = matrix defined by Eq (187) or Eq (219)

$\underline{\underline{C}}_j$ = trigonometric matrix defined by Eq (178)

C_k = scalar defined by Eq (83)

$\underline{\underline{C}}'_k$ = $\left\{ C'_x \mid C'_y \right\}_x$ = matrix of moments at a cross section of an edge beam

d_k = scalar defined by Eq (80)

$\underline{\underline{D}}$ = matrix defined by Eq (227)

$\underline{\underline{Q}}$ = trigonometric matrix defined by Eq (179)

D_n = rigidity coefficient for the stretching problem, given by Eq (18)

D_x, D_y = rigidity coefficients for the bending problem, given by Eqs (42) and (43)

$\underline{\underline{E}}_i$ = matrix defined by Eq (57)

E_k = Young's modulus for an edge beam

E_x, E_y = Young's moduli in the x and y directions for an orthotropic plate

$\underline{\underline{\bar{F}}}$ = matrix of forces applied externally to a rigid boundary portion

$\underset{\sim}{f}_{ij}$ = matrix dual of $\underset{\sim}{k}_{ij}$

f_{ss}, f_{zz} = elastic support flexibility coefficients

f_{xx}, f_{xy}, f_{yy} = edge beam flexibility coefficients given by Eqs (205) to (207)

$\underset{\sim}{\bar{F}}_i$ = (F_{xi}, F_{yi}) = vector of concentrated nodal forces

$\underset{\sim}{\Delta F}_i$ = matrix portion of edge loads resisted by an edge beam

$\underset{\sim}{\delta F}_k$ = matrix defined by Eq (123)

$\mathcal{F}(w)$ = scalar function of displacements (w) defined by Eq (148)

$\underset{\sim}{\mathcal{F}}_k$ = matrix defined by Eq (217)

G = shear modulus

G_i = scalar defined by Eq (57)

$\underset{\sim}{G}_{uk}$ = matrix defined by Eq (230)

$\underset{\sim}{G}_{zk}$ = matrix defined by Eq (231)

h_n = average thickness of an element

$\underset{\sim}{H}$ = matrix defined by Eq (220)

$\underset{\sim}{H}_n$ = matrix function of geometric properties of segment n, defined by Eq (94)

i,j,k = unit vectors in the x, y, and z directions

$\underset{\sim}{I}_2$ = (2x2) unit matrix

I_k = moment of inertia an edge beam about the z axis at node k

$\underset{\sim}{J}_n$ = matrix function of geometric properties of segment n, defined by Eq (94)

k_{xx}, k_{xy}, etc. = elastic support stiffness coefficients

k_{ij}^{xx}, k_{ij}^{xy}, etc. = edge beam stiffness factors, given by Eq (84)

$\underset{\sim}{K}$ = global stiffness supermatrix

$\underset{\sim}{K}'$ = global flexibility supermatrix, dual of $\underset{\sim}{K}$

K_x, K_y = particular bending solution functions

$\underset{\sim}{K}_{ij}^{n}$ = element stiffness submatrix, defined by Eqs (14) to (17) and Eq (24)

$\underset{\sim}{k}_{ij}$ = elastic support stiffness submatrix, defined by Eqs (69) to (72)

ℓ_i = length of segment i

$\underset{\sim}{L}_k$ = matrix defined by Eq (214)

$\bar{M}$ = vector of applied edge moments

$\underset{\sim}{M}_i$ = matrix of sines and cosines, defined by Eq (102)

M_p = moment applied externally to a point p

M_{nn} = edge moment (vector parallel to edge) magnitude

M_{xx}, M_{xy}, etc. = bending stress couples, defined by Fig. 5

$\bar{N}^i$ = (N_{nx}^i, N_{ny}^i) = vector of edge stress resultant intensity on edge i

N_{xi}, N_{yi} = nodal values of edge stress resultant intensity

$\bar{n}$ = unit outward normal vector to the boundary in the x-y plane

o = superscript denoting quantities associated with thermal effects

$\underset{\sim}{O}_2$ = (2×2) zero matrix

p = superscript denoting quantities associated with the particular solution of the bending problem

$\bar{p}$ = (p_x, p_y, p_z) = vector of distributed load intensity over the middle surface of the plate

P^n_{xi}, P^n_{yi} = generalized nodal load components at node i due to the distributed load components p_x and p_y over element n

$\underset{\sim}{P}'_i$ = matrix dual of $P_i = \left\{ P_{xi} \quad P_{yi} \right\}$, the matrix of generalized nodal loads due to p_x and p_y over all elements incident upon node i

Q = vector edge shear in the z direction

Q_{ne} = effective edge shear on the plate boundary in the z direction

$\underset{\sim}{Q}_i$ = matrix given by Eq (180)

Q_x, Q_y = shears on the x and y faces of a differential element, defined in Fig. 5

$\underset{\sim}{R}^i$ = rotation matrix given by Eq (58)

$\underset{\sim}{R}_i$ = generalized nodal load matrix due to edge load intensities

$\underset{\sim}{R}'_i$ = generalized nodal rotation matrix dual of $\underset{\sim}{R}_i$

R_{xi}, R_{yi} = components of $\underset{\sim}{R}_i$

s = distance along the boundary

s = superscript denoting quantities associated with the edge support

$\bar{s}$ = vector tangent to the boundary

s_k = scalar defined by Eq (82)

$\underset{\sim}{S}$ = column supermatrix of generalized nodal loads

$\underset{\sim}{S}'$ = column supermatrix of generalized nodal rotations, dual of $\underset{\sim}{S}$

$\underset{\sim}{S}^n_i$ = total generalized nodal force at node i due to element n

$\underset{\sim}{S}_i$ = total generalized nodal force matrix at node i due to all elements incident upon node i

$\underset{\sim}{T}_k$ = matrix defined by Eq (226)

ΔT = temperature change

$\bar{u}$	=	(u,v) = vector of in-plane displacement
u_i, v_i	=	x and y components of displacement of node or point i
$\underset{\sim}{U}$	=	column supermatrix of nodal displacements
$\underset{\sim}{U}'$	=	column supermatrix of nodal values of stress functions dual of $\underset{\sim}{U}$
U_i, V_i	=	values of stress functions at node i
$\underset{\sim}{U}_i'$	=	matrix of stress functions at node i, dual of $\underset{\sim}{U}_i = \{u_i \quad v_i\}$
$\underset{\sim}{U}_q^+, \underset{\sim}{U}_q^-$	=	matrix of displacements of a node q on a dislocation (+ and - side)
w	=	z component of displacement
$w_{,s}, w_{,n}$	=	slope of plate at edge in s and n directions
$w_{,x}, w_{,y}$	=	slope of plate at edge in x and y directions
W_n	=	rotation of segment n, given by Eq (78)
$\underset{\sim}{W}_k$	=	geometric matrix defined by Eq (150)
$\mathcal{W}_j$	=	scalar defined by Eq (181)
x,y,z	=	cartesian coordinates
x_i, y_i	=	coordinates of node or point i
X_p	=	distance from point p to node 1
Z_o	=	scalar defined by Eq (216)
α_x, α_y	=	coefficients of thermal expansion
α_k	=	scalar defined by Eq (81)
$\delta u^o, \delta v^o, \delta w^o$	=	components of the closing of a dislocation
ϵ_i	=	axial strain in segment i
$\underset{\sim}{\epsilon}_i$	=	column matrix defined by Eq (95)

$\epsilon_x^o, \epsilon_y^o$ = thermal strains in an edge beam

ζ_i = super-row i of the global stiffness matrix

$\underset{\sim}{\theta}_i$ = $\left\{\theta_{xi} \quad \theta_{yi}\right\}$ = generalized nodal force due to thermal effects, at node i, given by Eqs (21) and (22)

$\underset{\sim}{\theta}_i^!$ = generalized nodal rotation matrix due to thermal effects, dual of $\underset{\sim}{\theta}_i$

θ = angle defined by Fig. 11

ν_x, ν_y = Poisson's coefficients

π_n = potential energy of element n

ϕ_n = angle from x-axis to outward normal n on boundary

$\underset{\sim}{\Phi}_i$ = matrix defined by Eq (116)

$\times_i$ = boundary curvature deformation at node i

$\underset{\sim}{\times}_i$ = matrix defined by Eq (116)

ω_1 = specified rotation of segment 1

Ω_1 = bending problem quantity dual of ω_1

$\sum_k$ = symbol meaning "sum on all nodes k around the entire closed boundary path"

$\$_k$ = matrix defined by Eq (229)

$*$ = superscript denoting quantities associated with the homogeneous bending solution

$\Rightarrow$ = symbol meaning "is replaced by"

APPENDIX 5

List of Figures

APPENDIX 6

List of Tables

APPENDIX 7

PROGRAM LISTINGS

A7-1 CDL PROGRAMS

```
ADD 'BOUNDARY I'
EXECUTE 'BDINIT'
REPEAT TABULAR
DATA CHECK SET 'I10'
CONDITION 'I10' LT 1
NEW COMMAND
OTHERWISE
CALL 'MTRAN'
MOVE DOUBLE 'D10' TO 'D1'
END CONDITION
DATA CHECK SET 'I10'
CALL 'MTRAN'
MOVE DOUBLE 'D10' TO 'D2'
EXECUTE 'NBDASS'
END REPEAT TABULAR
FILE
```

```
REPLACE 'MTRAN'   $ A CDL SUBROUTINE
CONDITION 'I10' EQ 1
NO ID ALPHA 8 'D10'
OR CONDITION 'I10' EQ 3
NO ID INTEGER 'I36'
EXECUTE 'MTRANS'
OTHERWISE
MESSAGE '   '
MESSAGE 'INCORRECT NAME SPECIFICATION'
MESSAGE '   '
MESSAGE 'NODE,ELEMENT, AND BOUNDARY NAMES MUST BE'
MESSAGE 'INTEGER OR ALPHANUMERIC'
MESSAGE '   '
MESSAGE 'REMAINDER OF CURRENT COMMAND WILL BE SCANNED'
MESSAGE 'BUT NO EXECUTION OF SUBROUTINES WILL OCCUR.'
MESSAGE '   '
INHIBIT
END CONDITION
RETURN
FILE
```

```
ADD 'BOUNDARY C'
DATA CHECK SET 'I10'
CALL 'MTRAN'
MOVE DOUBLE 'D10' TO 'D1'
```

```
EXISTENCE 'DISPL' 'STRES' 'ELAST' 'EDGE' 'MIXED' 'RIGID' SET 'I1' REQUIR
IGNORE 'BEAM'
CONDITION 'I1' LE 2
   EXISTENCE 'S' 'B' 'G' SET 'I2'
OR CONDITION 'I1' EQ 5
   EXISTENCE 'STR' 'BEND/1' 'BEND/2' SET 'I2'
OR CONDITION 'I1' EQ 6
   EXISTENCE 'FI' 'PI' 'FR'  SET 'I2'
   NEXT RECORD
   ID 'U1' REAL 'T1' STANDARD 0.
   ID 'V1' REAL 'T2' STANDARD 0.
   ID 'R1' REAL 'T3' STANDARD 0.
   ID 'YP' REAL 'T4' STANDARD 0.
   ID 'MP' REAL 'T5' STANDARD 0.
   ID 'UP' REAL 'T1' STANDARD 0.
   ID 'VP' REAL 'T2' STANDARD 0.
   ID 'XP' REAL 'T3' STANDARD 0.
   ID 'FX' REAL 'T1' STANDARD 0.
   ID 'FY' REAL 'T2' STANDARD 0.
END CONDITION OPTIONAL
REPEAT TABULAR
DATA CHECK SET 'I10'
CONDITION 'I10' LT 1
NEW COMMAND
OTHERWISE
CALL 'MTRAN'
MOVE DOUBLE 'D10' TO 'D2'
END CONDITION
MODIFIER 'TO'
DATA CHECK SET 'I10'
CALL 'MTRAN'
MOVE DOUBLE 'D10' TO 'D3'
OTHERWISE
PRESET DOUBLE 'D3' EQUAL 0.
END MODIFIER
ID 'U' REAL 'T1'   STANDARD 0.
ID 'V' REAL 'T2'   STANDARD 0.
ID 'KXX' REAL 'T3' STANDARD 0.
ID 'KXY' REAL 'T4' STANDARD 0.
ID 'KYX' REAL 'T5' STANDARD 0.
ID 'KYY' REAL 'T6' STANDARD 0.
ID 'EPS' REAL 'T7' STANDARD 0.
ID 'W' REAL 'T8'   STANDARD 0.
ID 'R' REAL 'T9'  STANDARD 0.
ID 'IN' REAL 'T10'  STANDARD 0.
ID 'J' REAL 'T11'   STANDARD 0.
ID 'G' REAL 'T12'   STANDARD 0.
ID 'NX' REAL 'T1'   STANDARD 0.
ID 'NY' REAL 'T2'   STANDARD 0.
ID 'EB' REAL 'T3'   STANDARD 0.
ID 'IZ' REAL 'T4'   STANDARD 0.
ID 'NR' REAL 'T2'   STANDARD 0.
ID 'ANG' REAL 'T3'  STANDARD 0.
ID 'CHI' REAL 'T6'  STANDARD 0.
ID 'Q' REAL 'T8'    STANDARD 0.
```

```
ID 'MN' REAL 'T9'    STANDARD 0.
ID 'KZZ' REAL 'T10' STANDARD 0.
ID 'KSS' REAL 'T11' STANDARD 0.
EXECUTE 'NBCCP'
END REPEAT TABULAR
FILE


ADD 'STABC'    $ A CDL SUBROUTINE
DATA CHECK SET 'I10'
CALL 'MTRAN'
MOVE DOUBLE 'D10' TO 'D1'
DATA CHECK SET 'I10'
CALL 'MTRAN'
MOVE DOUBLE 'D10' TO 'D2'
MODIFIER 'TO'
   DATA CHECK SET 'I10'
   CALL 'MTRAN'
   MOVE DOUBLE 'D10' TO 'D3'
OTHERWISE
   PRESET DOUBLE 'D3' EQUAL 0.
END MODIFIER
EXECUTE 'NSCCP'
RETURN
FILE


ADD 'SIMPLE S'
PRESET INTEGER 'I1' EQUAL 1
CALL 'STABC'
FILE
ADD 'IN-PLANE R'
PRESET INTEGER 'I1' EQUAL 2
CALL 'STABC'
FILE
ADD 'NORMAL R'
PRESET INTEGER 'I1' EQUAL 3
CALL 'STABC'
FILE
ADD 'CLAMPED E'
PRESET INTEGER 'I1' EQUAL 4
CALL 'STABC'
FILE
ADD 'FREE E'
PRESET INTEGER 'I1' EQUAL 5
CALL 'STABC'
FILE


ADD 'LINE L'
PRESET 'I4' EQUAL C
REPEAT TABULAR
DATA CHECK SET 'I10'
CONDITION 'I10' LE 0
NEW COMMAND
```

```
OTHERWISE
CALL 'MTRAN'
MOVE DOUBLE 'D10' TO 'D2'
END CONDITION
IGNORE 'FOR'
ID 'X' REAL 'T1'
ID 'Y' REAL 'T2'
EXECUTE 'LINLOD'
END REPEAT TABULAR
FILE


REPLACE 'DISLOC'
PRESET 'I2' EQUAL 0
PRESET 'I4' EQUAL 0
ID 'DU' REAL 'T1' STANDARD 0.
ID 'DV' REAL 'T2' STANDARD 0.
ID 'ROT' REAL 'T3' STANDARD 0.
EXECUTE 'DISLCP'
NEXT RECORD
MODIFIER 'PATH'
REPEAT
DATA CHECK SET 'I10'
CONDITION 'I10' EQ -1
NEXT RECORD
NEW COMMAND
OR CONDITION 'I10' GE 0
CALL 'MTRAN'
INCREMENT 'I4'
EXECUTE 'DISLCP'
END CONDITION
END REPEAT
OTHERWISE
MESSAGE 'YOU FORGOT THE WORD -PATH-'
MESSAGE 'THE EFFECTS OF THE DISLOCATION WILL'
MESSAGE 'NOT BE INCLUDED IN THE PROBLEM'
MESSAGE 'BUT PROCESSING WILL CONTINUE.'
END MODIFIER
FILE


REPLACE 'GRAVITY L'
ID 'ANGLE-X' REAL 'T1' STANDARD 0.
ID 'ANGLE-Z' REAL 'T2' STANDARD 0.
EXECUTE 'GRVLOD'
FILE


ADD 'LPROC'  $ A CDL SUBROUTINE
DATA CHECK SET 'I10'
CALL 'MTRAN'
MOVE DOUBLE 'D10' TO 'D1'
PRESET INTEGER 'I3' EQUAL 0
CALL 'XROUT'
CONDITION 'I2' EQ 1
```

```
RETURN
END CONDITION OPTIONAL
MOVE DOUBLE 'D10' TO  'D2'
CALL 'XROUT'
CONDITION 'I2' EQ 1
RETURN
END CONDITION OPTIONAL
MOVE DOUBLE 'D10' TO  'D3'
CALL 'XROUT'
CONDITION 'I2' EQ 1
RETURN
END CONDITION OPTIONAL
MOVE DOUBLE 'D10' TO  'D4'
CALL 'XROUT'
CONDITION 'I2' EQ 1
RETURN
END CONDITION OPTIONAL
MOVE DOUBLE 'D10' TO  'D5'
CALL 'XROUT'
CONDITION 'I2' EQ 1
RETURN
END CONDITION OPTIONAL
MOVE DOUBLE 'D10' TO  'D6'
CALL 'XROUT'
CONDITION 'I2' EQ 1
RETURN
END CONDITION OPTIONAL
MOVE DOUBLE 'D10' TO  'D7'
CALL 'XROUT'
CONDITION 'I2' EQ 1
RETURN
END CONDITION OPTIONAL
MOVE DOUBLE 'D10' TO  'D8'
CALL 'XROUT'
CONDITION 'I2' EQ 1
RETURN
END CONDITION OPTIONAL
MOVE DOUBLE 'D10' TO  'D9'
CALL 'XROUT'
CONDITION 'I2' EQ 1
RETURN
END CONDITION OPTIONAL
PRESET INTEGER 'I3' EQUAL 10
RETURN
FILE


ADD 'XROUT'          $ A CDL SUBROUTINE
INCREMENT 'I3'
DATA CHECK SET 'I10'
CONDITION 'I10' EQ 2
PRESET 'I2' EQUAL 1
OR CONDITION 'I10' EQ 0
PRESET 'I2' EQUAL 1
OTHERWISE
```

```
PRESET 'I2' EQUAL 0
CALL 'MTRAN'
END CONDITION
RETURN
FILE


REPLACE 'BENDING PAR'
IGNORE 'SOL'
MODIFIER 'UNIFORM'
   ID 'L' REAL 'T1'  REQUIRED
   ID 'C' REAL 'T2' STANDARD 0.50
EXECUTE 'STDPSL'
OTHERWISE
REPEAT TABULAR 'END'
MODIFIER 'NO'  $ MODIFIER LEVEL IS 2
PRESET INTEGER 'I1' EQUAL 0
CALL 'LPROC'
ID 'KX' REAL 'T1' STANDARD 0.
ID 'KY' REAL 'T2' STANDARD 0.
   EXECUTE 'PARTIC'
OR MODIFIER 'EL'
PRESET INTEGER 'I1' EQUAL 1
CALL 'LPROC'
ID 'KX' REAL 'T1' STANDARD 0.
ID 'KY' REAL 'T2' STANDARD 0.
EXECUTE 'PARTIC'
OTHERWISE
MESSAGE '  '
MESSAGE ' PERHAPS YOU FORGOT THE END STATEMENT'
MESSAGE 'REMAINING COMMANDS WILL BE SCANNED BUT NO PROBLEM  '
MESSAGE 'EXECUTION WILL OCCUR'
MESSAGE '  '
INHIBIT
NEXT RECORD
NEW COMMAND
END MODIFIER
END REPEAT TABULAR
END MODIFIER
FILE
```

A7-2 INPUT PHASE PROGRAMS (ICETRAN AND FORTRAN IV)

```
      SUBROUTINE BDINIT
C
C    AUTHOR-     D. A. NAGY
C    LANGUAGE-   ICETRAN
C
C    THIS SUBROUTINE INITIALIZES THE ARRAY OF BOUNDARY NAMES,
C    BDID, THE POINTER ARRAY FOR BOUNDARY CONDITIONS, BDCOND,
C    AND THE PROBLEM TYPE INDICATOR IPROB.
C
      COMMON FILL1(182),ISCAN,FILL2(140),ID,FILL3(89),IPROB,IBCON
      COMMON BDID(P), BDCOND(P)
      DYNAMIC ARRAY BDID(D),BDCOND
      IBCON=0
      DEFINE BDID,1,DOUBLE,STEP=1
      DEFINE BDCOND,1,POINTER,STEP=1
      GO TO (7,7,7,7,7,7,7,7,7,1,3,5),ID
 1    IPROB=0
      RETURN
 3    IPROB=-1
      RETURN
 5    IPROB=2
      RETURN
C
C    IF ID IS NOT 10,11, OR 12, THE COMMAND 'BOUNDARY INCI-
C    DENCES,' WHICH CALLED THIS SUBROUTINE, IS INVALID AND
C    THE APPROPRIATE ERROR MESSAGE IS PRINTED.
C
 7    WRITE(6,8)
      WRITE(6,9)
      ISCAN=2
      RETURN
 8    FORMAT(51H THIS COMMAND VALID ONLY FOR DUAL PLATE STRETCHING,)
 9    FORMAT(51H DUAL PLATE BENDING, AND DUAL PLATE GENERAL--ERROR.)
      END
```

```
      DUM=JTID(NTEMP)
      WRITE (6,103) DUM
      GO TO 70
 7    IF(M-4) 8,9,70
 9    M=1
 8    IF(JTYP(ELTOP(I,M+1))) 40,40,41
 40   DUM=JTID(ELTOP(I,M+1))
      WRITE (6,100) DUM
      GO TO 70
 41   INDIC=JTYP(ELTOP(I,M+1))
      GO TO (12,13,13,70,70,70,70,12,12,13,13,70,70,70,70),INDIC
 12   J=I+1
      DO 30 I=J,NBXTEL
      DO 30 M=2,4
      IF(ELTOP(I,M)-NTEMP) 30,7,30
 30   CONTINUE
C
C    ERROR IF AT ANY POINT THE CHAIN IS BROKEN BY SOME NODE
C    THAT WAS NOT SPECIFIED AS A BOUNDARY NODE WHEN IT
C    SHOULD HAVE BEEN.
C
      DUM=JTID(NTEMP)
      WRITE (6,102) DUM
      GO TO 70
 13   NFIRST=ELTOP(I,M+1)
      J=I+1
 190  DO 50 IB=J,NBXTEL
      DO 50 MB=2,4
      IF(ELTOP(IB,MB)-NTEMP) 50,51,50
 50   CONTINUE
      NTEMP=NFIRST
      GO TO 31
 51   IF(MB-4) 52,53,70
 53   MB=1
 52   IF(JTYP(ELTOP(IB,MB+1))) 54,54,55
 54   I=IB
      M=MB
      GO TO 40
 55   INDIC=JTYP(ELTOP(IB,MB+1))
      GO TO (62,63,63,70,70,70,70,62,62,63,63,70,70,70,70),INDIC
 62   J=IB+1
      GO TO 190
 63   N2ND=ELTOP(IB,MB+1))
      GO TO (80,81,82),M
 80   IF(ELTOP(I,3)-N2ND) 83,84,83
 84   NTEMP=NFIRST
      GO TO 31
 83   GO TO (85,86,87),MB
 81   IF(ELTOP(I,4)-N2ND) 83,84,83
 82   IF(ELTOP(I,2)-N2ND) 83,84,83
 85   IF(ELTOP(IB,3)-NFIRST) 88,89,88
 86   IF(ELTOP(IB,4)-NFIRST) 88,89,88
 87   IF(ELTOP(IB,2)-NFIRST) 88,89,88
 89   NTEMP=N2ND
      GO TO 31
```

```
      SUBROUTINE NBDASS
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE ACTS ON THE TABULAR REPEATS OF THE 'BOUNDARY
C     INCIDENCES' COMMAND, INCREMENTING THE CONNECTIVITY COUNTER
C     IBCON AND CONSTRUCTING THE CHAIN OF BOUNDARY NODES, IN
C     ORDER AROUND THE BOUNDARY IN THE +S DIRECTION.
C     THIS SUBROUTINE ASSUMES THAT NO ELEMENT TOUCHES MORE THAN ONE
C     CLOSED BOUNDARY CURVE.
C
      COMMON FILL1(72),D1,D2,FILL2(106),ISCAN,FILL3(78),JTID(P),DUMM
      COMMON JEXTN,JTYP(P),FILL4(98),ELTOP(P),FILL5(16),NBXTEL,FILL6(30)
      COMMON IBCON, BDID(P), BDCOND(P)
      DOUBLE PRECISION D1,D2,DUM
      DYNAMIC ARRAY BDID(0),BDCOND,JTYP,JTID(0),ELTOP(I)
      IBCON=IBCON+1
      N=0
      BDID(IBCON)=D1
      DEFINE BDCOND(IBCON),10,POINTER,STEP=10
C
C        LOCATE NODE NUMBER OF FIRST NODE
C
      DO 1 I=1,JEXTN
      IF(LOBLE(D2,JTID(I))) 1,2,1
 1    CONTINUE
C
C     ERROR IF THE NODE NAMED BY D2  WAS NOT PREVIOUSLY
C     DEFINED.
C
      WRITE(6,100) D2
 7    ISCAN=2
      RETURN
 2    IF(JTYP(I)-3) 3,4,5
C
C     ERROR IF THE  NODE NAMED BY D2 IS NOT ACTUALLY ON A
C     BOUNDARY (SPECIFIED BY BOUNDARY OR B IN NODE COORDINATES
C     COMMAND).
C
 3    WRITE(6,101) D2
      GO TO 70
 5    IF(JTYP(I)-11) 3,4,4
 4    NTEMP=I
 10   N=N+1
      DEFINE BDCOND(IBCON,N),5,FULL,STEP=5
      DESTROY BDCOND(IBCON,N)
      BDCOND(IBCON,N,1)=NTEMP
C
C        LOCATE NEXT NODE ALONG BOUNDARY IN +S DIRECTION
C
      DO 20 I=1,NBXTEL
      DO 20 M=2,4
      IF(ELTOP(I,M)-NTEMP) 20,7,20
 20   CONTINUE
```

```
      GO TO (90,91,92),M
90    INDIC=JTYP(ELTOP(I,3))
      GO TO 93
91    INDIC=JTYP(ELTOP(I,4))
      GO TO 93
92    INDIC=JTYP(ELTOP(I,2))
93    GO TO (94,95,95,70,70,70,70,94,94,95,95,70,70,70,70),INDIC
94    NTEMP=NFIRST
      GO TO 31
95    NTEMP=N2ND
31    NNN=BDCOND(IBCON,1,1)
      IF(NTEMP-NNN) 10,180,10
180   BDCOND(IBCON,N+1,1)=0.
      RELEASE BDID
      RELEASE BDCOND
      RELEASE JTYP
      RELEASE JTID
      RELEASE ELTOP
      RETURN
100   FORMAT(6H NODE ,A8, 36H  WAS NOT PREVIOUSLY DEFINED--ERROR.)
101   FORMAT(6H NODE ,A8, 30H  IS NOT ON A BOUNDARY--ERROR.)
102   FORMAT(28H NODE IMMEDIATELY FOLLOWING ,A8, 44H  WAS NOT SPECIFIED
     1AS BOUNDARY NODE--ERROR.)
103   FORMAT(6H NODE ,A8,37H  NOT INCIDENT ON ANY ELEMENT--ERROR.)
      END
```

```
      SUBROUTINE MTRANS
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   FORTRAN IV
C
C     THIS SUBROUTINE TRANSFORMS AN INTEGER NUMBER OF UP TO
C     FOUR DIGITS FROM INTEGER FORMAT TO ALPHANUMERIC FORMAT,
C     LEFT-ADJUSTED IN THE 8-CHARACTER FIELD.
C
      COMMON DUMM(60),IFILL(35),I36,FILL(54),D10
      DOUBLE PRECISION DNAME,D10
      DIMENSION I(2)
      EQUIVALENCE (I(1),DNAME)
C
C     ADD 4 BLANK CHARACTERS TO I(2)--EBCDIC REPRESENTATION
C
      I(2)=1073741824
C
C     DETERMINE FIRST DIGIT OF THE NUMBER
C
      DO 1 J=1,4
      IFD=I36/((10**(5-J))/10)
      IF(IFD) 1,1,2
    1 CONTINUE
      RETURN
C
C     ADD EBCDIC CODE FOR FIRST DIGIT TO LEFTMOST CHARACTER
C     POSITION OF I(1).
C
    2 I(1)=-268435456+(2**24)*IFD
      K=J
      KK=4-J
      IF(J-4) 4,3,3
C
C     DETERMINE REMAINING DIGITS OF THE NUMBER AND STORE
C     THEIR EBCDIC CODES IN SUCCESSIVE CHARACTER POSITIONS
C     OF I(1).
C
    4 DO 15 N=1,KK
      I36=I36-IFD*10**(5-K)/10
      IFD=I36/((10**(4-K))/10)
      I(1)=I(1)+(2**(8*(4-N))*15)/16+(2**(8*(4-N))*IFD)/(2**8)
   13 K=K+1
      GO TO 15
    3 N=0
   15 IF(N-3) 20,21,21
C
C     ADD BLANK CHARACTERS TO REMAINING POSITIONS OF I(1)
C     NOT FILLED BY DIGIT CODES.
C
   20 M=N+1
      DO 14 L=M,3
   14 I(1)=I(1)+2**((4-N)*8)/4
C
C     TRANSFER RESULTING ALPHANUMERIC REPRESENTATION OF THE
```

```
C     NUMBER TO VARIABLE D10.
C
 21     D10=DNAME
        RETURN
        END
```

```
      SUBROUTINE NBCCP
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE PROCESSES THE 'BOUNDARY CONDITIONS' COMMAND.
C
      DOUBLE PRECISION D1, D2, D3
      DIMENSION DATA(14)
      COMMON I1,I2,IFILL(32),DATA,IFILL(24),D1,D2,D3,FILL1(98)
      COMMON FILL2(59),CFLEN,CFWT,CFANG,FILL3(175),IPROB,FILL4(3)
      COMMON BDCOND(P)
      DYNAMIC ARRAY BDCOND, JTLOB(R)
C
C     DETERMINE BOUNDARY NUMBER, POSITIONS OF THE NODE(S)
C     IN THE CHAIN, AND THE TYPES OF BOUNDARY CONDITIONS
C     BEFORE AND AFTER CURRENT PORTION.
C
      CALL GETNOS(D1,D2,D3,I,J,K,JC,KC)
      DATA(1)=0.
      DATA(2)=0.
      IFF=BDCOND(I,JC,2)+1.
      ITF=BDCOND(I,JC,3)+1.
      IF(KC) 79,80,79
   79 IL=BDCOND(I,KC,2)+1.
      ITL=BDCOND(I,KC,3)+1.
   80 IF(IPROB) 1,2,2
    1 DO 3 M=1,5
    3 DATA(M+2)=DATA(M+9)
C
C     BRANCH TO APPROPRIATE ROUTINE FOR STORING SPECIFIED BOUNDARY
C     CONDITION VALUES, DEPENDING ON THE VALUE OF I1, THE
C     BOUNDARY CONDITION TYPE INDICATOR SET BY CDL.
C
    2 GO TO (10,20,30,40,50,60),I1
C
C     DISPLACEMENT B.C.'S ROUTINE
C
   10 IF(IPROB) 11,12,13
   11 DATA(2)=1.
      DATA(3)=DATA(3)/CFLEN
      DATA(4)=DATA(4)/CFANG
      IF(KC) 70,70,71
   70 KC=JC
      GO TO 72
   71 CALL NSTBV(I,JC,KC-1,4,DATA,4)
   72 CALL NSTBV(I,KC,KC,5,DATA,4)
   99 RELEASE BDCOND
      DO 150 I=1,14
  150 DATA(I)=0.
      RETURN
   12 DATA(1)=1.
      DATA(3)=DATA(3)/CFLEN
      DATA(4)=DATA(4)/CFLEN
```

```
      IF(KC) 73,73,74
   73 KC=JC
      GO TO 75
   74 CALL NSTBV(I,JC,KC-1,4,DATA,0)
   75 CALL NSTBV(I,KC,KC,4,DATA,0)
  102 N=JC
  101 NODE=BDCOND(I,N,1)
      JTLOD(NODE,1,7)=DATA(3)
      JTLOD(NODE,1,8)=DATA(4)
      IF(KC-N) 98,99,98
   98 N=N+1
      IF(BDCOND(I,N,1)) 100,100,101
  100 N=1
      GO TO 101
   13 GO TO (12,14,15),I2
   15 DATA(1)=1.
   14 DATA(2)=1.
      DATA(3)=DATA(3)/CFLEN
      DATA(4)=DATA(4)/CFLEN
      DATA(10)=DATA(10)/CFLEN
      DATA(11)=DATA(11)/CFANG
      IF(KC) 76,77,76
   77 KC=JC
      GO TO 78
   76 CALL NSTBV(I,JC,KC-1,11,DATA,0)
   78 CALL NSTBV(I,KC,KC,10,DATA,0)
      GO TO (102,99,102),I2
C
C   STRESS B.C.'S ROUTINE
C
   20 IF(IPROB) 21,22,23
   21 DATA(2)=2.
      IE=4
      DATA(3)=DATA(3)*CFLEN/CFWT
      DATA(4)=DATA(4)/CFWT
   84 IF(KC) 83,83,81
   83 DATA(IE)=0.
      GO TO 24
   81 CALL NSTBV(I,JC+1,KC-1,IE,DATA,4)
      GO TO (25,24,25,25,25,24,25),ITL
   25 CALL NSTBV(I,KC,KC,IE-1,DATA,4)
   24 GO TO (27,26,27,27,27,26,27),ITF
   26 DATA(IE-1)=0.
      DATA(2)=0.
   27 CALL NSTBV(I,JC,JC,IE,DATA,4)
      GO TO 99
   22 DATA(1)=2.
      DATA(3)=DATA(3)*CFLEN/CFWT
      DATA(4)=DATA(4)*CFLEN/CFWT
      IF(KC) 29,29,82
   82 CALL NSTBV(I,JC+1,KC-1,4,DATA,4)
      IF(IL-1) 28,28,29
   28 CALL NSTBV(I,KC,KC,4,DATA,4)
   29 GO TO (16,17,16,16,16,17,16,16,16),IFF
   16 CALL NSTBV(I,JC,JC,4,DATA,4)
```

```
 17   IF(IPROB) 99,99,19
 23   GO TO (22,18,22),I2
 18   DATA(3)=0.
      DATA(4)=0.
      DATA(9)=2.
      DATA(10)=DATA(10)*CFLEN/CFWT
      DATA(11)=DATA(11)/CFWT
      IE=11
      GO TO 84
 19   IF(I2-2) 99,99,18
C
C    ELASTIC BOUNDARY ROUTINE
C
 31   IF(IPROB) 31,33,32
 32   IEND=13
 161  DATA(3)=DATA(3)/CFLEN
      DATA(4)=DATA(4)/CFLEN
      DO 160 I=5,8
 160  DATA(I)=DATA(I)*CFLEN*CFLEN/CFWT
      DATA(10)=DATA(10)/CFLEN
      DATA(11)=DATA(11)/CFANG
      DATA(12)=DATA(12)*CFLEN*CFLEN/CFWT
      DATA(13)=DATA(13)*CFANG/CFWT
      GO TO 35
 33   IEND=8
      GO TO 161
 31   IEND=6
      DATA(3)=DATA(3)/CFLEN
      DATA(4)=DATA(4)/CFANG
      DATA(5)=DATA(5)*CFLEN*CFLEN/CFWT
      DATA(6)=DATA(6)*CFANG/CFWT
 35   ICHNG=4
      DATA(1)=I1
      DATA(2)=I1
      IF(KC) 89,99,89
 9    KC=JC
      GO TO 91
 89   CALL NSTBV(I,JC,KC-1,IEND,DATA,ICHNG)
 91   CALL NSTBV(I,KC,KC,IEND,DATA,ICHNG)
      GO TO 99
C
C    EDGE BEAM ROUTINE
C
 41   IF(IPROB) 41,43,42
 42   IEND=14
      N=10
 162  DATA(3)=DATA(3)*CFLEN/CFWT
      DATA(4)=DATA(4)*CFLEN/CFWT
      DATA(5)=DATA(5)*CFLEN*CFLEN/CFWT
      DATA(6)=DATA(6)/(CFLEN**4.)
 163  DATA(N)=DATA(N)*CFLEN/CFWT
      DATA(N+1)=DATA(N+1)/CFWT
      DATA(N+2)=DATA(N+2)/(CFLEN**4.)
      DATA(N+3)=DATA(N+3)/(CFLEN**4.)
      DATA(N+4)=DATA(N+4)*CFLEN*CFLEN/CFWT
```

```
      GO TO 35
 43   IEND=6
      [redacted]
 41   IEND=7
      N=3
      GO TO 163
C
C   MIXED BOUNDARY CONDITIONS ROUTINE
C
 50   GO TO (51,52,53),I2
 51   DATA(1)=5.
      DATA(3)=DATA(3)/CFLEN
      DATA(4)=DATA(4)*CFLEN/CFWT
      DATA(5)=DATA(5)/CFANG
      IF(KC) 85,85,86
 85   KC=JC
      IL=IFF
      GO TO 55
 86   CALL NSTBV(I,JC+1,KC-1,5,DATA,4)
      IF(IFF-2) 54,55,54
 54   CALL NSTBV(I,JC,JC,5,DATA,4)
 55   GO TO (56,99,56,56,56,56,56,99,56),IL
 56   CALL NSTBV(I,KC,KC,5,DATA,4)
      GO TO 99
 52   DATA(2)=5.
      IF(IPROB) 58,58,59
 58   N=3
      IE=4
 164  DATA(N)=DATA(N)/CFLEN
      DATA(N+1)=DATA(N+1)*CFLEN/CFWT
      IF(KC) 87,87,88
 87   KC=JC
      GO TO 92
 88   CALL NSTBV(I,JC,KC-1,IE,DATA,4)
 92   CALL NSTBV(I,KC,KC,IE-1,DATA,4)
      GO TO 99
 59   N=10
      IE=11
      GO TO 164
 53   DATA(2)=6.
      IF(IPROB) 5,5,6
 5    IL=4
      N=3
 165  DATA(N)=DATA(N)*CFLEN/CFWT
      DATA(N+1)=DATA(N+1)/CFANG
      IF(KC) 93,94,93
 94   DATA(IL)=0.
      GO TO 7
 93   CALL NSTBV(I,JC+1,KC-1,IE,DATA,4)
      GO TO (8,7,8,7,7,7,8),ITL
 8    CALL NSTBV(I,KC,KC,IE-1,DATA,4)
 7    IF(ITF-1) 66,66,9
 9    DATA(IE-1)=0.
 66   CALL NSTBV(I,JC,JC,IE,DATA,4)
      GO TO 99
```

```
 6      IE=11
        N=10
        GO TO 165
C
C    RIGID BOUNDARY ROUTINE
C
 60     DATA(6)=DATA(6)/CFLEN
        DATA(7)=DATA(7)/(CFLEN*CFWT)
        DATA(8)=DATA(8)*CFLEN
        GO TO (61,62,166),I2
 61     IE=9
        DATA(3)=DATA(3)/CFLEN
        DATA(4)=DATA(4)/CFLEN
        DATA(5)=DATA(5)/CFANG
        IF(IL-1) 65,65,64
 63     IF(KC) 95,95,96
 65     KC=JC+1
        IL=IE-1
        GO TO 64
 95     CALL NSTBV(I,KC,KC,IE,DATA,4)
 64     CALL NSTBV(I,JC,KC-1,IE+1,DATA,4)
        GO TO 99
 62     IE=8
        DATA(3)=DATA(3)/CFLEN
        DATA(4)=DATA(4)/CFLEN
 167    DATA(5)=DATA(5)/CFLEN
        GO TO 65
 166    IE=8
        DATA(3)=DATA(3)/CFWT
        DATA(4)=DATA(4)/CFWT
        GO TO 167
        END
```

```
      SUBROUTINE GETNOS(D1,D2,D3,I,J,K,JC,KC)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE OBTAINS THE BOUNDARY NUMBER AND POSI-
C     TIONS OF THE FIRST (AND LAST) NODE OF THE SPECIFIED
C     BOUNDARY PORTION, GIVEN THE BOUNDARY NAME(D1), AND NODE
C     NAMES (D2 AND D3).
C
      DOUBLE PRECISION D1, D2, D3
      COMMON FILL1(182),ISCAN,FILL2(78),JTID(P),DUM,JEXTN,FILL3(149)
      COMMON IBCON, BDID(P), BDCOND(P)
      DYNAMIC ARRAY BDID(D), JTID(D), BDCOND
      DO 1 I=1,IBCON
      IF(LCOBLE(D1,BDID(I))) 1,2,1
 1    CONTINUE
C
C        ERROR--BOUNDARY NOT PREVIOUSLY DEFINED
C
      WRITE(6,100) D1
 90   ISCAN=2
      GO TO 12
C
C        I NOW EQUALS BOUNDARY NUMBER
C
 2    DO 3 J=1,JEXTN
      IF(LCOBLE(D2,JTID(J))) 3,4,3
 3    CONTINUE
C
C        ERROR--NODE-1 NOT PREVIOUSLY DEFINED
C
      WRITE(6,101) D2
      GO TO 90
C
C        J NOW EQUALS NUMBER OF NODE-1
C
 4    DO 5 K=1,JEXTN
      IF(LCOBLE(D3, JTID(K))) 5,6,5
 5    CONTINUE
      IF(D3) 15,16,15
 16   K=0
      GO TO 6
C
C        ERROR--NODE-2 NOT PREVIOUSLY DEFINED
C
 15   WRITE(6,101) D3
      GO TO 90
C
C        K NOW EQUALS NUMBER OF NODE-2
C
 6    JC=1
      FJ=J
 9    IF(BDCOND(I,JC,1)-FJ) 7,8,7
 7    JC=JC+1
```

```
      IF(BDCOND(I,JC,1)) 10,10,9
C
C        ERROR--NODE-1 NOT ON GIVEN BOUNDARY
C
 10   WRITE(6,102) D2, D1
      GO TO 90
C
C        JC NOW EQUALS POSIGION OF NODE-1 IN CHAIN
C
  8   IF(K) 17,18,17
 18   KC=0
      GO TO 12
 17   KC=1
      FK=K
 12   IF(BDCOND(I,KC,1)-FK) 11,12,11
 11   KC=KC+1
      IF(BDCOND(I,KC,1)) 14,14,13
C
C        ERROR--NODE-2 NOT ON GIVEN BOUNDARY
C
 14   D2=D3
      GO TO 10
 12   RELEASE BDID
      RELEASE JTID
      RELEASE BDCOND
      RETURN
 101  FORMAT(10H BOUNDARY ,A8,32H  NOT PREVIOUSLY DEFINED--ERROR.)
 111  FORMAT(6H NODE ,A8,32H  NOT PREVIOUSLY DEFINED--ERROR.)
 102  FORMAT(6H NODE ,A8,18H  NOT ON BOUNDARY ,A8,9H --ERROR.)
      END
```

```
      SUBROUTINE NSTBV(IB,JP,KP,IEND,ARRAY,ICHNG)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE STORES THE FIRST IEND VALUES OF THE
C     DIMENSIONED ARRAY 'ARRAY' INTO THE DYNAMIC ARRAY
C     BDCOND FOR BOUNDARY NUMBER=IBN FROM NODE POSITION JP
C     TO NODE POSITION KP.  IT ALSO ADDS TO THE CODE IN
C     ARRAY JTYP TO INDICATE THAT THE BOUNDARY CONDITION
C     IS OR IS NOT ACTUALLY FULL DISPLACEMENT RESTRAINT
C     AT THE GIVEN NODE.       THIS IS DONE BY ADDING THE
C     INTEGER 4 TO JTYP IF THE BOUNDARY CONDITION IS
C     NOT THE DISPLACEMENT TYPE (FOR THE STRETCHING PROBLEM),
C     OR ADDING  0 IF IT IS.  THE 4 OR 0 IS PASSED TO SUB-
C     ROUTINE NSTBV THROUGH THE VARIABLE ICHNG.
C
      COMMON FILL1(265),JTYP(P),FILL2(150), BDCOND(P)
      DYNAMIC ARRAY BDCOND, JTYP
      IN=JP
 5    DO 10 I=1,IEND
      IF(ARRAY(I)) 1,10,1
 1    BDCOND(IB,IN,I+1)=ARRAY(I)
 10   CONTINUE
      INODE=BDCOND(IB,IN,1)
      JTYP(INODE)=JTYP(INODE)+ICHNG
      IF(IN-KP) 2,3,2
 2    IN=IN+1
      IF(BDCOND(IB,IN,1)) 5,4,5
 4    IN=1
      GO TO 5
 3    RELEASE BDCOND
      RELEASE JTYP
      RETURN
      END
```

```
      SUBROUTINE NSCCP
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   FORTRAN IV
C
C     THIS SUBROUTINE PROCESSES THE STANDARD SUPPORT COMMANDS.
C     IT BRANCHES TO THE APPROPRIATE ROUTINE ON THE
C     INDICATOR I1, SETS THE INDICATORS I1 AND I2 FOR
C     THE APPROPRIATE BOUNDARY CONDITION TYPES IMPLIED
C     BY THE USER, AND CALLS NBCCP.
      COMMON QQDUB(2), ICOM, IERROR, ICOML, QQCOM(75)
      COMMON I1,I2,FILL1(32),DATA(11),FILL2(368),IPROB
      DO 1 I=1,11
 1    DATA(I)=0.
      GO TO (10,20,30,40,50),I1
 10   IF(IPROB) 11,12,13
 11   I1=5
      I2=2
      CALL NBCCP
      RETURN
 13   I1=5
      I2=2
      CALL NBCCP
      I2=1
 12   I1=1
      CALL NBCCP
      RETURN
 20   IF(IPROB) 11,22,23
 23   I1=5
      I2=2
      CALL NBCCP
      I2=1
 22   I1=2
      CALL NBCCP
      RETURN
 30   IF(IPROB) 31,12,32
 31   I1=2
      CALL NBCCP
      RETURN
 32   I1=2
      I2=2
      CALL NBCCP
      I2=1
      GO TO 12
 40   I1=1
      I2=3
      CALL NBCCP
      RETURN
 50   I1=2
      I2=3
      CALL NBCCP
      RETURN
      END
```

```
      SUBROUTINE LINLOD
C
C   AUTHOR-     D. A. NAGY
C   LANGUAGE-   ICETRAN
C
C   THIS SUBROUTINE COMPUTES THE EQUIVALENT CONCENTRATED
C   NODAL LOADS (X AND Y COMPONENTS ONLY) FROM THE
C   SPECIFIED LINE LOAD INTENSITIES.
C
      DOUBLE PRECISION D2
      COMMON I1,ILL,NPR,I4,FILL1(32),T1,T2,X(2),Y(2),FXPR,FYPR
      COMMON FILL6(30),D2,FILL2(106),ISCAN,FILL3(52),CFLEN,CFWT
      COMMON FILL4(16),LEXTN,FILL5(7),JTID(P),DUM,JEXTN,FILL7(5)
      COMMON JTXYZ(P),JTLOD(P),FILL8(91),ELTOP(P),FILL9(16),NBXTEL
      DYNAMIC ARRAY JTXYZ(R),JTID(D),ELTOP(I),JTLOD(R)
      ILL=0
      II=IDEF(JUNK,JUNK,JUNK,JTLOD)
      IF(II) 113,14,113
 14   DEFINE JTLOD,10,POINTER,STEP=10
 113  DO 1 I=1,JEXTN
      IF(LCDBLE(D2,JTID(I))) 1,2,1
 1    CONTINUE
C
C   ERROR IF A NODE ON THE PATH OF THE LINE LOAD WAS NOT
C   PREVIOUSLY DEFINED.
C
      WRITE(6,100) D2
      ISCAN=2
      ILL=1
      GO TO 120
 2    IF(I4) 3,3,4
 3    IF(ILL) 5,5,20
 5    II=IDEF(JUNK,JUNK,JUNK,JTLOD,I)
      IF(II) 17,18,17
 18   DEFINE JTLOD(I),5,POINTER,STEP=5
 20   DEFINE JTLOD(I,LEXTN),12,FULL
      GO TO 19
 17   II=IDEF(JUNK,JUNK,JUNK,JTLOD,I,LEXTN)
      IF(II) 19,20,19
 19   X(1)=JTXYZ(I,1)
      Y(1)=JTXYZ(I,2)
 13   FXPR=T1*CFLEN/CFWT
      FYPR=T2*CFLEN/CFWT
      NPR=I
      I4=1
      RELEASE JTXYZ
      RELEASE JTLOD
 121  RELEASE ELTOP
 120  RELEASE JTID
      RETURN
 4    N=1
 10   DO 9 JE=N,NBXTEL
      DO 9 K=2,4
      IF(ELTOP(JE,K)-I) 9,8,9
 9    CONTINUE
```

```
C
C    ERROR IF THE LINE LOAD CUTS ACROSS AN ELEMENT AT ANY
C    POINT--IT MUST COINCIDE WITH ELEMENT EDGES AT ALL TIMES.
C
      WRITE(6,101) D2
      ISCAN=2
      ILL=1
      GO TO 121
 8    DO 11 L=2,4
      IF(ELTOP(JE,L)-NPR) 11,12,11
 11   CONTINUE
      N=JE+1
      GO TO 10
 12   X(2)=JTXYZ(I,1)
      Y(2)=JTXYZ(I,2)
      FLNGTH=((X(1)-X(2))**2.+(Y(1)-Y(2))**2.)**.5
      RXP=FLNGTH/6.*(2.*FXPR+T1*CFLEN/CFWT)
      RYP=FLNGTH/6.*(2.*FYPR+T2*CFLEN/CFWT)
      RXC=FLNGTH/6.*(2.*T1*CFLEN/CFWT+FXPR)
      RYC=FLNGTH/6.*(2.*T2*CFLEN/CFWT+FYPR)
      II=IDEF(JUNK,JUNK,JUNK,JTLOD,I)
      IF(II) 21,22,21
 22   DEFINE JTLOD(I),5,POINTER,STEP=5
      GO TO 16
 21   II=IDEF(JUNK,JUNK,JUNK,JTLOD,I,LEXTN)
      IF(II) 15,16,15
 16   DEFINE JTLOD(I,LEXTN),12,FULL
 15   JTLOD(NPR,LEXTN,1)=JTLOD(NPR,LEXTN,1)+RXP
      JTLOD(NPR,LEXTN,2)=JTLOD(NPR,LEXTN,2)+RYP
      JTLOD(I,LEXTN,1)=JTLOD(I,LEXTN,1)+RXC
      JTLOD(I,LEXTN,2)=JTLOD(I,LEXTN,2)+RYC
      X(1)=X(2)
      Y(1)=Y(2)
      GO TO 13
 100  FORMAT(6H NODE ,A8,32H  NOT PREVIOUSLY DEFINED--ERROR.)
 101 0FORMAT(6H NODE ,A8,55H  NOT CONNECTED TO PREVIOUS NODE BY ANY ELEM
     1ENT--ERROR.)
      END
```

```
      SUBROUTINE GRVLOD
C
C   AUTHOR-     D. A. NAGY
C   LANGUAGE-   ICETRAN
C
C   THIS SUBROUTINE COMPUTES THE GRAVITY LOAD (X AND
C   Y COMPONENTS ONLY) ON EACH ELEMENT, DIVIDES IT INTO
C   THIRDS, AND ADDS THE RESULTING CONTRIBUTION
C   TO THE NODAL LOADS ARRAY JTLOD.
C
      DIMENSION X(3), Y(3), N(3)
      COMMON FILL1(36),T1,T2,FILL2(144),ISCAN,FILL3(54),CFANG,FILL10(15)
      COMMON LEXTN,FILL4(16),JTXYZ(P),JTLOD(P),FILL5(85),ELID(P)
      COMMON FILL6(2),ELPROP(P),ELTOP(P),FILL7(16),NBXTEL,FILL8(6)
      COMMON ELSTDS,FILL9(22),IPROB
      DOUBLE PRECISION DUM
      DYNAMIC ARRAY ELTOP,ELPROP,JTXYZ(R),JTLOD(R),ELID(D)
      DEFINE JTLOD,JEXTN,POINTER
      DO 200 I=1,JEXTN
      DEFINE JTLOD(I),LEXTN,POINTER,STEP=5
 200  DEFINE JTLOD(I,LEXTN),12,FULL
      IF(IPROB) 1,2,3
C
C   IF THE PROBLEM IS ONLY PLATE BENDING, A MESSAGE IS
C   PRINTED AND THE SUBROUTINE IS NOT EXECUTED ANY FURTHER.
C
 1    WRITE(6,100)
      WRITE(6,104)
      WRITE(6,101)
      RETURN
 5    DUM=ELID(I)
      WRITE (6,103) DUM
      ISCAN=2
      RETURN
 3    WRITE(6,102)
      WRITE(6,100)
      WRITE(6,104)
 2    IF(T2) 7,1,7
 7    T1=T1/CFANG
      T2=T2/CFANG
C
C   LOOP ON ALL ELEMENTS
C
      DO 10 I=1,NBXTEL
      DO 4 J=2,4
      IF(ELTOP(I,J)) 5,5,4
 4    CONTINUE
      DO 6 J=1,3
      N(J)=ELTOP(I,J+1)
      X(J)=JTXYZ(N(J),1)
 6    Y(J)=JTXYZ(N(J),2)
      IF(ELPROP(I,13)) 12,11,12
 11   ELPROP(I,13)=ELSTDS
C
C   COMPUTE ELEMENT AREA, OBTAIN ELEMENT DENSITY AND
```

```
C    THICKNESS, AND COMPUTE FORCE COMPONENTS
C
 12   OAREA=.5*ABS(-X(1)*(-Y(2)+Y(3))-X(2)*(-Y(3)+Y(1))-X(3)*(-Y(1)+Y(2))
      1)
       FX=-AREA*ELPROP(I,13)*ELPROP(I,2)*SIN(T2)/3.
       FY=-SIN(T1)*FX
       FX=-COS(T1)*FX
C
C    ADD RESULTS TO JTLOD FOR 3 NODES INCIDENT UPON CURRENT ELEMENT.
C
       DO 10 J=1,3
       JTLOD(N(J),LEXTN,1)=JTLOD(N(J),LEXTN,1)+FX
       JTLOD(N(J),LEXTN,2)=JTLOD(N(J),LEXTN,2)+FY
 10    CONTINUE
       RETURN
 100   FORMAT(49H COMPONENT OF GRAVITY LOAD PERPENDICULAR TO PLATE)
 104   FORMAT(51H MUST BE SPECIFIED VIA PARTICULAR BENDING SOLUTION.)
 101   FORMAT(33H GRAVITY LOAD COMMAND IS IGNORED.)
 102   FORMAT(10H REMINDER-)
 103   FORMAT(9H ELEMENT ,A8,39H  NODE INCIDENCE INCORRECTLY SPECIFIED.)
       END
```

```
      SUBROUTINE DISLCP
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE PROCESSES THE 'DISLOCATION' COMMAND.
      DOUBLE PRECISION D2
      COMMON I1,I2,I3,I4,FILL1(32),T1,T2,T3,FILL2(35),D2,FILL3(106)
      COMMON ISCAN,FILL4(52),CFLEN,CFWT,CFANG,FILL7(23),JTID(P),DUM
      COMMON JEXTN,FILL5(148),IPROB,IBCON,FILL6(4),DISLOC(P)
      DYNAMIC ARRAY DISLOC, JTID(D)
      IF(I2-1) 1,2,11
C
C     IF THE PROBLEM IS PLATE BENDING ONLY,OR IF THE
C     PLATE IS SIMPLY CONNECTED, A MESSAGE IS PRINTED OUT
C     AND THE COMMAND IS IGNORED.
C
 1    IF(IBCON-1) 14,14,15
 14   WRITE(6,103)
      WRITE(6,101)
      I2=2
      RETURN
 15   IF(IPROB) 9,10,10
 9    WRITE(6,100)
      WRITE(6,101)
      I2=2
 11   RETURN
 10   II=IDEF(JUNK,JUNK,JUNK,DISLOC)
      IF(II) 3,3,4
 3    DEFINE DISLOC,5,POINTER,STEP=1
      I=1
      GO TO 5
 4    I=1
 7    II=IDEF(JUNK,JUNK,JUNK,DISLOC,I)
      IF(II) 5,5,6
 5    DEFINE DISLOC(I),10,FULL,STEP=10
      GO TO 8
 6    I=I+1
      GO TO 7
C
C     THE VALUES OF THE RIGID BODY CLOSING OF THE DISLOC)
C     ARE CONVERTED TO INTERNAL UNITS AND STORED.
C
 8    DISLOC(I,1)=T1/CFLEN
      DISLOC(I,2)=T2/CFLEN
      DISLOC(I,3)=T3/CFANG
      I2=1
      I3=I
      RELEASE DISLOC
      RETURN
 2    DO 12 J=1,JEXTN
      IF(LCDBLE(D2,JTID(J))) 12,13,12
 12   CONTINUE
C
C     ERROR IF A NODE ON THE PATH OF THE DISLOCATION WAS
```

```
C    NOT PREVIOUSLY DEFINED.
C
       WRITE(6,102) D2
       ISCAN=2
       RETURN
C
C    STORE NUMBER OF NODE ON THE PATH OF THE DISLOCATION
C
 13    DISLOC(I3,I4+3)=J
       RELEASE JTID
       RELEASE DISLOC
       RETURN
 100 0FORMAT(62H DISLOCATION COMMAND APPLIES ONLY TO THE PLANE STRESS PR
     1OBLEM.)
 101   FORMAT(25H COMMAND WILL BE IGNORED.)
 102   FORMAT(6H NODE ,A8,36H  WAS NOT PREVIOUSLY DEFINED--ERROR.)
 103 0FORMAT(63H DISLOCATION COMMAND APPLIES ONLY TO MULTIPLY-CONNECTED
     1PLATES.)
       END
```

```
      SUBROUTINE PARTIC
C
C    AUTHOR-     D. A. NAGY
C    LANGUAGE-   ICETRAN
C
C    THIS SUBROUTINE STORES THE SPECIFIED NODAL OR ELEMENT-
C    CENTERED VALUES OF THE PARTICULAR SOLUTION IN THE
C    ARRAYS PBSOLN (FOR NODAL VALUES) AND PBSOLE (FOR
C    ELEMENT VALUES)
      COMMON I1,I2,I3,FILL1(33),T1,T2,FILL2(34),DNAME(10),FILL3(90)
      COMMON ISCAN
      COMMON FILL4(52),CFLEN,FILL5(17),LEXTN,FILL6(7),JTID(P),DUM
      COMMON JEXTN,FILL7(94),ELID(P),FILL8(22),NBXTEL,FILL9(37)
      COMMON PBSOLN(P),PBSOLE(P),FILL10(4),IPRTIC(P)
      DOUBLE PRECISION DNAME
      DYNAMIC ARRAY PBSOLN,PBSOLE,JTID(D),ELID(D),IPRTIC
      II=IDEF(JUNK,JUNK,JUNK,IPRTIC)
      WRITE (6,900) LEXTN
      IF(II) 13,14,13
 14   DEFINE IPRTIC,LEXTN,HALF,STEP=5
 13   IPRTIC(LEXTN)=0
      IF(I1) 1,1,2
 1    II=IDEF(JUNK,JUNK,JUNK,PBSOLN)
      IF(II) 3,3,4
 3    DEFINE PBSOLN,LEXTN,POINTER,STEP=5
      GO TO 16
 4    II=IDEF(JUNK,JUNK,JUNK,PBSOLN,LEXTN)
      IF(II) 15,16,15
 16   DEFINE PBSOLN(LEXTN),JEXTN,POINTER
      DO 17 I=1,JEXTN
 17   DEFINE PBSOLN(LEXTN,I),2,FULL
      DESTROY PBSOLN(LEXTN)
 15   DO 5 I=1,I3
      DO 6 J=1,JEXTN
      IF(LCDBLE(DNAME(I),JTID(J))) 6,7,6
 6    CONTINUE
C
C    ERROR IF A GIVEN NODE WAS NOT PREVIOUSLY DEFINED.
C
      WRITE(6,100) DNAME(I)
      ISCAN=2
      RETURN
 7    PBSOLN(LEXTN,J,1)=T1*CFLEN
 5    PBSOLN(LEXTN,J,2)=T2*CFLEN
      RELEASE PBSOLN
      RETURN
 2    II=IDEF(JUNK,JUNK,JUNK,PBSOLE)
      IF(II) 8,8,18
 8    DEFINE PBSOLE,LEXTN,POINTER,STEP=5
      GO TO 19
 9    II=IDEF(JUNK,JUNK,JUNK,PBSOLE,LEXTN)
      IF(II) 18,19,18
 19   DEFINE PBSOLE(LEXTN),JEXTN,POINTER
      DO 20 I=1,NBXTEL
 20   DEFINE PBSOLE(LEXTN,I),2,FULL
```

```
      DESTROY PBSOLE(LEXTN)
 18   DO 10 I=1,I3
      DO 11 J=1,NBXTEL
      IF(LCDBLE(DNAME(I),ELID(J))) 11,12,11
 11   CONTINUE
C
C    ERROR IF A GIVEN ELEMENT WAS NOT PREVIOUSLY DEFINED.
C
      WRITE(6,101) DNAME(I)
      ISCAN=2
      RETURN
 12   PBSOLE(LEXTN,J,1)=T1*CFLEN
 10   PBSOLE(LEXTN,J,2)=T2*CFLEN
      RELEASE PBSOLE
      RETURN
 100  FORMAT(6H NODE ,A8,32H  NOT PREVIOUSLY DEFINED--ERROR.)
 101  FORMAT(9H ELEMENT ,A8,32H  NOT PREVIOUSLY DEFINED--ERROR.)
 900  FORMAT(7H LEXTN=, I2)
      END
```

```
      SUBROUTINE STDPSL
C
C   AUTHOR-     D. A. NAGY
C   LANGUAGE-   ICETRAN
C
C   THIS SUBROUTINE COMPUTES THE EXACT INTEGRALS OF
C   THE STANDARD PARTICULAR SOLUTION OVER EACH ELEMENT
C   AND THE EXACT NODAL VALUES FOR EACH NODE.
C
      DIMENSION X(3), Y(3)
      COMMON FILL1(36),T1,T2,FULL2(144),ISCAN,FILL3(52),CFLEN,CFWT
      COMMON FILL4(16),LEXTN,FILL5(10),JEXTN,FILL6(5),JTXYZ(P)
      COMMON FILL7(87),ELID(P),DUM1(2),ELPROP(P),ELTOP(P),FILL8(16)
      COMMON NBXTEL,FILL9(3),ELSTDE,FILL10(3),ELSTPO,FILL11(29)
      COMMON PBSOLN(P),PBSOLE(P),FILL12(4),IPRTIC(P)
      DOUBLE PRECISION DUM
      DYNAMIC ARRAY PBSOLN,PBSOLE,JTXYZ(R),ELTOP(I),ELPROP,ELID(D)
      DYNAMIC ARRAY IPRTIC
      PI=3.1415926
      C=T2
      P=-T1*CFLEN*CFLEN/CFWT
      II=IDEF(JUNK,JUNK,JUNK,IPRTIC)
      WRITE (6,900) LEXTN
      IF(II) 22,23,22
 23   DEFINE IPRTIC,5,HALF,STEP=5
 22   .IPRTIC(LEXTN)=1
C
C        OBTAIN MATERIAL PROPERTIES
C
C   COMPUTATIONS ASSUME A HOMOGENEOUS PLATE--THUS ALL
C   PROPERTIES ARE OBTAINED FROM THE FIRST ELEMENT.
C
      EX=ELPROP(1,6)
      IF(EX) 1,2,1
 2    EX=ELSTDE
 1    EY=ELPROP(1,7)
      IF(EY) 3,4,3
 4    EY=EX
 3    PX=ELPROP(1,8)
      IF(PX) 5,6,5
 6    PX=ELSTPO
 5    PY=EX*PX/EY
      H=ELPROP(1,2)
      IF(H) 7,8,7
 8    DUM=ELID(1)
      WRITE (6,100) DUM
      ISCAN=2
      RETURN
C
C   DEFINE PARTICULAR SOLUTION STORAGE ARRAYS PBSOLN AND
C   PBSOLE.
C
 7    II=IDEF(JUNK,JUNK,JUNK,PBSOLN)
      IF(II) 24,25,24
 25   DEFINE PBSOLN,LEXTN,POINTER,STEP=5
```

```
 24    DEFINE PBSOLN(LEXTN),JEXTN,POINTER
       DO 26 I=1,JEXTN
 26    DEFINE PBSOLN(LEXTN,I),2,FULL
       II=IDEF(JUNK,JUNK,JUNK,PBSOLE)
       IF(II) 27,28,27
 28    DEFINE PBSOLE,LEXTN,POINTER,STEP=5
 27    DEFINE PBSOLE(LEXTN),JEXTN,POINTER
       DO 29 I=1,JEXTN
 29    DEFINE PBSOLE(LEXTN,I),2,FULL
C
C         DETERMINE AXES FOR PARTICULAR SOLUTION
C         FIND MAXIMUM AND MINIMUM X AND Y COORDINATES
C
       XMAX=JTXYZ(1,1)
       XMIN=XMAX
       YMAX=JTXYZ(1,2)
       YMIN=YMAX
       DO 15 I=2,JEXTN
       IF(XMAX-JTXYZ(I,1)) 10,9,9
 10    XMAX=JTXYZ(I,1)
       GO TO 11
 9     IF(XMIN-JTXYZ(I,1)) 11,11,12
 12    XMIN=JTXYZ(I,1)
 11    IF(YMAX-JTXYZ(I,2)) 13,14,14
 13    YMAX=JTXYZ(I,2)
       GO TO 15
 14    IF(YMIN-JTXYZ(I,2)) 15,15,16
 16    YMIN=JTXYZ(I,2)
 15    CONTINUE
       A=ABS((XMAX-XMIN)/2.)
       B=ABS((YMAX-YMIN)/2.)
C
C         COMPUTE AND STORE NODAL VALUES OF PARTICULAR SOLUTION
C
       DO 116 I=1,JEXTN
       XG=JTXYZ(I,1)
       YG=JTXYZ(I,2)
       XX=XG-XMIN-A
       YY=YG-YMIN-B
       FKX=6.*P/(EY*H**3.)*((1.-C)*(YY*YY-B*B)-C*PX*(XX*XX-A*A))
       FKY=6.*P/(EY*H**3.)*(C*(XX*XX-A*A)-PX*(1.-C)*(YY*YY-B*B))
       PBSOLN(LEXTN,I,1)=FKX
 116   PBSOLN(LEXTN,I,2)=FKY
C
C         COMPUTE AND STORE INTEGRALS OF THE PARTICULAR SOLUTION
C         OVER EACH ELEMENT SURFACE.
C
       DO 17 I=1,NBXTEL
       DO 18 J=1,3
       X(J)=JTXYZ(ELTOP(I,J+1),1)
 18    Y(J)=JTXYZ(ELTOP(I,J+1),2)
       B1=Y(3)-Y(2)
       B2=Y(1)-Y(3)
       B3=Y(2)-Y(1)
       AREA=0.5*ABS(-X(1)*B1-X(2)*B2-X(3)*B3)
```

```
      IF(X(2)-X(1)) 19,20,19
 20   THETA=PI/2.
      XB=ABS(Y(2)-Y(1))
      GO TO 21
 19   THETA=ATAN((Y(2)-Y(1))/(X(2)-X(1)))
      XB=(X(2)-X(1))/COS(THETA)
 21   YBAR=2.*AREA/XB
      FL13S=(X(3)-X(1))**2.+((Y(3)-Y(1))**2.)
      XBAR=(FL13S-YBAR**2.)**0.5
      SIT=SIN(THETA)
      COST=COS(THETA)
      C1=X(1)-XMIN-A
      C2=Y(1)-YMIN-B
      D=6.*P/(EY*H**3.)
      SINSQ=SIT**2.
      COSSQ=COST*COST
      FX2=XB*YBAR*(XB*XB+XB*XBAR+XBAR*XBAR)/12.0
      FY2=(XB*YBAR**3.)/12.0
      FXY=XB*YBAR**2.*(XB+2.*XBAR)/24.0
      FX1=XB*YBAR*(XB+XBAR)/6.0
      FY1=YBAR*YBAR*XB/6.0
      FCON=XB*YBAR/2.0
C
C        COMPUTE INTEGRAL OF KX OVER SURFACE
C
      FKX=FX2*((1.-C)*SINSQ-C*PX*COSSQ)
      FKX=FKX+FY2*((1.-C)*COSSQ-C*PX*SINSQ)
      FKX=FKX+FXY*2.*SIT*COST*(1.-C+C*PX)
      FKX=FKX+FX1*2.*((1.-C)*C2*SIT-C*PX*C1*COST)
      FKX=FKX+FY1*2.*((1.-C)*C2*COST-C*PX*C1*SIT)
      FKX=FKX+FCON*((1.-C)*(C2*C2-B*B)-C*PX*(C1*C1-A*A))
      FKX=FKX*D
      PBSOLE(LEXTN,I,1)=FKX
C
C        COMPUTE INTEGRAL OF KY OVER ELEMENT SURFACE
C
      FKY=FX2*((C*COSSQ-PX*(1.-C)*SINSQ))
      FKY=FKY+FY2*(C*SINSQ-PX*(1.-C)*COSSQ)
      FKY=FKY+FXY*2.*SIT*COST*(C-PX*(1.-C))
      FKY=FKY+FX1*2.*(C*C1*COST-PX*(1.-C)*SIT)
      FKY=FKY+FY1*2.*(C*C1*SIT-PX*(1.-C)*COST)
      FKY=FKY+FCON*(C*(C1*C1-A*A)-PX*(1.-C)*(C2*C2-B*B))
      FKY=D*FKY
 17   PBSOLE(LEXTN,I,2)=FKY
      RELEASE PBSOLE
      RELEASE PBSOLN
      RELEASE ELPROP
      RELEASE ELTOP
      RELEASE JTXYZ
      RELEASE ELID
      RELEASE IPRTIC
      RETURN
 100  FORMAT(9H ELEMENT ,A8, 28H  HAS ZERO THICKNESS--ERROR.)
 900  FORMAT(7H LEXTN=, I2)
      END
```

A7-3 ELEMENT STIFFNESS/FLEXIBILITY MATRIX GENERATION PHASE.

```
      SUBROUTINE STNGEN
C
C   AUTHOR-     D. A. NAGY
C   LANGUAGE-   ICETRAN
C
C   THIS SUBROUTINE IS THE EXECUTIVE FOR THE ELEMENT
C   STIFFNESS/FLEXIBILITY MATRIX GENERATION PHASE.
C
      COMMON FILL1(182),ISCAN,FILL2(87),JTXYZ(P),FILL3(87),ELID(P)
      COMMON FILL4(2),ELPROP(P),ELTOP(P),FILL5(14),ELEXT(P),FILL6
      COMMON NBEL,FILL7(2),ELSTDE,ELSTDG,FILL8(2),ELSTPO,FILL9(4)
      COMMON ELSTMT(P),FILL10(15),IPROB
      DYNAMIC ARRAY JTXYZ(R),ELID(D),ELPROP,ELTOP(I),ELEXT(I),ELSTMT
      DIMENSION TEMP(24), NODE(3)
      DOUBLE PRECISION DUM
      TYPE='FTOD'
      DEFINE ELSTMT,NBEL,6,4
C
C   LOOP ON ALL ACTIVE ELEMENTS
      DO 7 I=1,NBEL
      IN=ELEXT(I)
      IF(TYPE-ELPROP(ELEXT(I),1)) 1,2,1
C
C        ERROR--WRONG ELEMENT TYPE
C
 1    DUM=ELID(IN)
      WRITE (6,100) DUM
      ISCAN=2
      RELEASE ELID
      GO TO 99
C
C   DETERMINE COORDINATES OF ELEMENTS THREE VERTICES,
C
 2    DO 3 J=1,3
 3    NODE(J)=ELTOP(ELEXT(I),J+1)
      X1=JTXYZ(NODE(1),1)
      Y1=JTXYZ(NODE(1),2)
      X2=JTXYZ(NODE(2),1)
      Y2=JTXYZ(NODE(2),2)
      X3=JTXYZ(NODE(3),1)
      Y3=JTXYZ(NODE(3),2)
C
C   RETRIEVE THE ELEMENT PROPERTIES.
C
      EX=ELPROP(IN,6)
      IF(EX) 20,21,20
 21   EX=ELSTDE
 20   EY=ELPROP(IN,7)
      PX=ELPROP(IN,8)
      IF(PX) 22,23,22
 23   PX=ELSTPO
```

```
 22   PY=PX*EX/EY
      G=ELPROP(IN,12)
      IF(G) 24,25,24
 25   G=ELSTDG
 24   H=ELPROP(IN,2)
      IF(H) 5,4,5
C
C        ERROR--ELEMENT HAS ZERO THICKNESS
C
 4    DUM=ELID(IN)
      WRITE (6,101) DUM
      ISCAN=2
      RELEASE ELID
      GO TO 6
C
C   CALL STGFTD TO COMPUTE THE DIAGONAL AND LOWER HALF
C   OF THE MATRIX FOR THE CURRENT ELEMENT.  ANSWER IS
C   RETURNED IN THE ARRAY TEMP AND MUST BE STORED IN
C   THE DYNAMIC ARRAY ELSTMT.
C
 5    CALL STGFTD(X1,Y1,X2,Y2,X3,Y3,EX,EY,PX,PY,G,H,IPROB,TEMP)
      DO 7 J=1,6
      DO 7 K=1,4
      MA=4*J+K-4
 7    ELSTMT(I,J,K)=TEMP(MA)
 6    RELEASE JTXYZ
      RELEASE ELTOP
 99   RELEASE ELSTMT
      RELEASE ELPROP
      RELEASE ELEXT
      RETURN
 100  FORMAT(9H ELEMENT ,A8,31H  IS NOT OF TYPE 'FTOD'--ERROR.)
 101  FORMAT(9H ELEMENT ,A8,28H  HAS ZERO THICKNESS--ERROR.)
      END
```

```
      SUBROUTINE STGFTD(X1,Y1,X2,Y2,X3,Y3,EX,EY,PX,PY,G,H,IPROB,TEMP)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   FORTRAN IV
C
C     THIS SUBROUTINE COMPUTES THE DIAGONAL AND LOWER HALF
C     OF THE ELEMENT STIFFNESS/FLEXIBILITY MATRIX.  FOR THE
C     BENDING PROBLEM, IT CALLS NDUAL TO PERFORM THE
C     DUALITY CONVERSION OF PROPERTIES.
C
      DIMENSION A(3), B(3), TEMP(24)
      IF(EY) 1,1,2
 1    EY=EX
 2    IF(PY) 5,5,4
 4    IF(EX*PX-EY*PY) 5,6,5
 5    PY=EX*PX/EY
 6    IF(IPROB) 7,8,8
 7    CALL NDUAL(EX,EY,PX,PY,G,H)
 8    A(1)=X3-X2
      A(2)=X1-X3
      A(3)=X2-X1
      B(1)=Y3-Y2
      B(2)=Y1-Y3
      B(3)=Y2-Y1
      AREA=0.5*ABS(-X1*B(1)-X2*B(2)-X3*B(3))
      D=H/(4.0*AREA*(1.-PX*PY))
      N=1
      PC=(1.-PX*PY)*G
      DO 9 I=1,3
      DO 9 J=1,I
      TEMP(N)=D*(EX*B(I)*B(J)+PC*A(I)*A(J))
      TEMP(N+1)=-D*(EX*PX*B(I)*A(J)+PC*A(I)*B(J))
      TEMP(N+2)=-D*(EY*PY*A(I)*B(J)+PC*B(I)*A(J))
      TEMP(N+3)=D*(EY*A(I)*A(J)+PC*B(I)*B(J))
      N=N+4
 9    CONTINUE
      RETURN
      END


      SUBROUTINE NDUAL(EX,EY,PX,PY,G,H)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   FORTRAN IV
C
C     THIS SUBROUTINE PERFORMS THE DUALITY CONVERSION
C
      EXT=-12.0*(1.0-PX*PY)/(EY*H**4.0)
      EY=EXT*EY/EX
      G=-3.0/(G*H**4.0)
      PX=-PX
      PY=-PY
      EX=EXT
      RETURN
      END
```

A7-4 NON-SYMMETRIC STRUCTURAL STIFFNESS/FLEXIBILITY MATRIX GENERATION PHASE.

```
      SUBROUTINE STNSAS
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE GENERATES THE NON-SYMMETRIC STIFFNESS/
C     FLEXIBILITY MATRIX.  IT IS STORED IN DYNAMIC ARRAY FCMAT.
C
      COMMON FILL1(100),NSOL,FILL2(67),FCMAT(P),ICUREL(P),IREL1(P)
      COMMON FILL3(155)
      COMMON JINT(P),FILL4(34),ELTOP(P),FILL5(14),ELEXT(P),NDUM,NBEL
      COMMON FILL6(11),ELSTMT(P)
      DYNAMIC ARRAY FCMAT(R),ICUREL,IREL1,ELSTMT,ELEXT(I),JINT,ELTOP(I)
      DIMENSION NODE(3)
C
C     DEFINE STORAGE ARRAY FCMAT AND BOOKKEEPING ARRAYS
C     ICUREL AND IREL1.
C
      DEFINE FCMAT,NJ,POINTER
      DEFINE ICUREL,NJ,POINTER
      DEFINE IREL1,2*NJ,NJ,HALF
      DESTROY IREL1
C
C     LOOP ON ALL ACTIVE ELEMENTS.
C
      DO 2 IE=1,NBEL
      DO 1 N=1,3
 1    NODE(N)=JINT(ELTOP(ELEXT(IE)))
      NS=0
      DO 2 I=1,3
      DO 2 J=1,I
      NS=NS+1
      IF(NODE(I)-NJ) 3,3,2
 3    IF(NODE(J)-NJ) 4,4,2
 4    DEFINE FCMAT(NODE(J)),2*NJ,POINTER
      II=IDEF(JNK,JNK,JNK,FCMAT,NODE(J),2*NODE(I))
      IF(II) 10,11,10
 11   DEFINE FCMAT(NODE(J),2*NODE(I)-1),2,FULL
      DESTROY FCMAT(NODE(J),2*NODE(I)-1)
      DEFINE FCMAT(NODE(J),2*NODE(I)),2,FULL
      DESTROY FCMAT(NODE(J),2*NODE(I))
C
C     ADD CONTRIBUTION OF CURRENT ELEMENT TO FCMAT
C
 10   OFCMAT(NODE(J),2*NODE(I)-1,1)=FCMAT(NODE(J),2*NODE(I)-1,1) +
     1 ELSTMT(IE,NS,1)
      OFCMAT(NODE(J),2*NODE(I)-1,2)=FCMAT(NODE(J),2*NODE(I)-1,2) +
     1 ELSTMT(IE,NS,2)
C
C     UPDATE BOOKKEEPING ARRAY IREL1
C
```

```
      IREL1(2*NODE(I)-1,NODE(J))=1
     OFCMAT(NODE(J),2*NODE(I),1)=FCMAT(NODE(J),2*NODE(I),1) +
     1 ELSTMT(IE,NS,3)
     OFCMAT(NODE(J),2*NODE(I),2)=FCMAT(NODE(J),2*NODE(I),2) +
     1 ELSTMT(IE,NS,4)
      IREL1(2*NODE(I),NODE(J))=1
      IF(I-J) 5,2,5
 5    DEFINE FCMAT(NODE(I)),2*NJ,POINTER
      II=IDEF(JNK,JNK,JNK,FCMAT,NODE(I),2*NODE(J))
      IF(II) 20,21,20
 21   DEFINE FCMAT(NODE(I),2*NODE(J)-1),2,FULL
      DESTROY FCMAT(NODE(I),2*NODE(J)-1)
      DEFINE FCMAT(NODE(I),2*NODE(J)),2,FULL
      DESTROY FCMAT(NODE(I),2*NODE(J))
 20   OFCMAT(NODE(I),2*NODE(J)-1,1)=FCMAT(NODE(I),2*NODE(J)-1,1) +
     1 ELSTMT(IE,NS,1)
      IREL1(2*NODE(J)-1,NODE(I))=1
     OFCMAT(NODE(I),2*NODE(J)-1,2)=FCMAT(NODE(I),2*NODE(J)-1,2) +
     1 ELSTMT(IE,NS,3)
     OFCMAT(NODE(I),2*NODE(J),1)=FCMAT(NODE(I),2*NODE(J),1) +
     1 ELSTMT(IE,NS,2)
     OFCMAT(NODE(I),2*NODE(J),2)=FCMAT(NODE(I),2*NODE(J),2) +
     1 ELSTMT(IE,NS,4)
      IREL1(2*NODE(J),NODE(I))=1
 2    CONTINUE
C
C    CONSTRUCT BOOKKEEPING ARRAY ICUREL
C
      DO 8 IC=1,NJ
      N=0
      DEFINE ICUREL(IC),5,HALF,STEP=5
      NJ2=2*NJ
      DO 6 IR=1,NJ2
      IF(FCMAT(IC,IR,1)) 7,6,7
 7    N=N+1
      ICUREL(IC,N)=IR
 6    CONTINUE
      DEFINE ICUREL(IC),N,HALF
 8    CONTINUE
      RELEASE FCMAT
      RELEASE ICUREL
      RELEASE IREL1
      RELEASE ELSTMT
      RELEASE ELEXT
      RELEASE JINT
      RELEASE ELTOP
C
C    ASSEMBLE STACK OF PROGRAMS FOR REMAINDER OF PROBLEM SOLUTION.
C
      ADD TO STACK (1,'STNBKS')
      ADD TO STACK (1,'STNSSL')
      ADD TO STACK (1,'STNBCM')
      TRANSFER TO STACK
      END
```

A7-5 BOUNDARY CONDITIONS PHASE

```
      SUBROUTINE STNBCM
C
C   AUTHOR-     D. A. NAGY
C   LANGUAGE-   ICETRAN
C
C   THIS IS THE EXECUTIVE PROGRAM FOR THE BOUNDARY
C   CONDITIONS PHASE.  IT PROCESSES THE BOUNDARY CON-
C   DITIONS FOR EACH BOUNDARY IN ORDER AROUND THE
C   BOUNDARY BY CALLING A DICTIONARY SUBROUTINE WHICH
C   IN TURN CALLS THE APPROPRIATE BOUNDARY CONDITION
C   SUBROUTINE.  IT ALSO GENERATES THE LOAD OR
C   ROTATION VECTOR.
C
      COMMON IMIN,FILL1(99),NSOL,FILL2(67),FCMAT(P),ICUREL(P),IREL1(P)
      COMMON FILL3(158),IPROB,IBCON,BDID(P),BDCOND(P)
      DOUBLE PRECISION DUM
      DYNAMIC ARRAY BDID(D),BDCOND
      DYNAMIC ARRAY FCMAT,ICUREL,IREL1
      IMIN=0
      IF(IPROB) 1,2,2
1     CALL STNBLV
      L=3
      GO TO 3
2     CALL STNSLV
      L=2
      II=IDEF(JNK,JNK,JNK,DISLOC)
      IF(II) 70,3,101
101   CALL SDISLC
      GO TO 3
3     DO 99 IBN=1,IBCON
      J=1
      NN=BDCOND(IBN,1,L)
      IF(NN) 20,51,20
51    DUM=BDID(IBN)
      WRITE(6,100) DUM
      GO TO 70
20    J=J+1
      NST=BDCOND(IBN,J,L)
      IF(NST) 50,51,50
50    IF(NN-NST) 10,11,10
11    IF(BDCOND(IBN,J+1,1)) 20,21,20
21    J=1
10    NODE=J
      GO TO (31,31,30),L
31    CALL DICTS(IBN,NST,J)
      GO TO 32
30    CALL DICTB(IBN,NST,J)
32    IF(J-NODE) 40,99,40
40    NST=BDCOND(IBN,J,L)
      GO TO 10
99    CONTINUE
      IF(IMIN-3) 200,201,201
```

```
200   WRITE (6,102)
      WRITE (6,103)
      WRITE (6,104)
      GO TO 70
201   GO TO 300
300   DEFINE FCMAT,NSOL,POINTER
      DEFINE ICUREL,NSOL,POINTER
      DEFINE IREL1,2*NSOL,NSOL,HALF
      DO 60 I=1,NSOL
      II=IDEF(JNK,JNK,JNK,FCMAT,I)
      IF(II) 61,60,61
61    DEFINE FCMAT(I),2*NSOL,POINTER
60    CONTINUE
      DO 64 IC=1,NSOL
      N=IDEF(JNK,JNK,JNK,ICUREL,IC)
      DO 62 I=1,N
      IF(ICUREL(IC,I)-NSOL) 62,62,63
62    CONTINUE
      I=N+1
63    DEFINE ICUREL(IC),I-1,HALF
64    CONTINUE
70    RELEASE BDID
      RELEASE BDCOND
      RETURN
100 0FORMAT(25H CONDITIONS FOR BOUNDARY ,A8, 32H NOT COMPLETELY SPECIFI
     1ED--ERROR)
102   FORMAT(50H ERROR--MINIMUM BOUNDARY CONDITIONS NOT SPECIFIED.)
103   FORMAT(52H PLEASE RESUBMIT PROBLEM AND SPECIFY AT LEAST 3 DIS-)
104   FORMAT(49H PLACEMENT COMPONENTS FOR THE STRETCHING PROBLEM.)
      END
```

```
      SUBROUTINE STNSLV
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE CONSTRUCTS THE GENERALIZED NODAL LOAD
C     VECTOR FOR THE STRETCHING PROBLEM.
C
      COMMON FILL1(100),NSOL,NN,NLDSI,FILL2(47),LEXT(P),FILL3(14)
      COMMON KPPRI(P),FILL4(104),JTLOD(P),FILL5(43),NJ,FILL6(11),JEXT(P)
      DYNAMIC ARRAY KPPRI(R),JEXT,LEXT,JTLOD(R)
      DEFINE KPPRI,NJ,2*NLDSI
      DESTROY KPPRI
C
C     ADD CONCENTRATED NODE LOADS TO LOAD VECTOR (INCLUDES LINE LOADS
C     AND GRAVITY LOADS)
C
      DO 99 I=1,NSOL
      NODE=JEXT(I)
      DO 99 L=1,NLDSI
      LOAD=LEXT(L)
      KPPRI(NODE,2*L-1)=KPPRI(NODE,2*L-1)+JTLOD(NODE,LOAD,1)
 99   KPPRI(NODE,2*L)=KPPRI(NODE,2*L)+JTLOD(NODE,LOAD,2)
C
C     ADD ELEMENT (SURFACE AND VOLUME) LOADS TO LOAD VECTOR
C
C     ADD THIS ROUTINE WHEN 'ELEMENT LOADS' COMMAND BECOMES
C     OPERATIONAL
C
C     ADD ELEMENT TEMPERATURE LOADS TO LOAD VECTOR
C
C     ADD THIS ROUTINE WHEN 'ELEMENT TEMPERATURE LOADS' COMMAND
C     BECOMES OPERATIONAL
C
      RELEASE KPPRI
      RELEASE JEXT
      RELEASE LEXT
      RELEASE JTLOD
      RETURN
      END
```

```
      SUBROUTINE STNBLV
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE ASSEMBLES THE GENERALIZED NODAL
C     ROTATION VECTOR FOR THE BENDING PROBLEM.
C
      COMMON FILL1(102),NLDSI,FILL2(47),LEXT(P),FILL3(14),KPPRI(P)
      COMMON FILL4(102),JTXYZ(P),FILL5(45),NJ,FILL6(9),JINT(P)
      COMMON FILL7(34),ELTOP(P),FILL8(14),ELEXT(P),NFIL,NBEL
      COMMON FILL9(46),PBSOLN(P),PBSOLE(P),FILL10(4),IPRTIC(P)
      DIMENSION A(3),B(3),NODE(3),X(3),Y(3)
      DYNAMIC ARRAY KPPRI(R),ELEXT(I),ELTOP(I),JTXYZ(R),JINT,LEXT
      DYNAMIC ARRAY IPRTIC,PBSOLE,PBSOLN
      DEFINE KPPRI,NJ,2*NLDSI
      DESTROY KPPRI
      DO 99 IE=1,NBEL
      NEL=ELEXT(IE)
      DO 1 N=1,3
      NODE(N)=ELTOP(NEL,N+1)
      NODE(N)=JINT(NODE(N))
      X(N)=JTXYZ(NODE(N),1)
1     Y(N)=JTXYZ(NODE(N),2)
      A(1)=X(3)-X(2)
      A(2)=X(1)-X(3)
      A(3)=X(2)-X(1)
      B(1)=Y(3)-Y(2)
      B(2)=Y(1)-Y(3)
      B(3)=Y(2)-Y(1)
      AREA=0.
      GX=0.
      GY=0.
      DO 2 N=1,3
2     AREA=AREA-X(N)*B(N)
      AREA=ABS(AREA)/2.
C
C     ADD CONTRIBUTION OF PARTICULAR SOLUTION TO GENERALIZED NODAL
C     ROTATION VECTOR
C
      DO 98 L=1,NLDSI
      LN=LEXT(L)
      IF(IPRTIC(LN)) 3,4,3
3     DO 5 N=1,3
      IF(NODE(N)-NJ) 7,7,5
7     OKPPRI(NODE(N),2*L-1)=KPPRI(NODE(N),2*L-1)+B(N)/(2.*AREA)*PBSOLE(
      1 LN,NEL,1)
      OKPPRI(NODE(N),2*L)=KPPRI(NODE(N),2*L)-A(N)/(2.*AREA)*PBSOLE(LN,
      1  NEL,2)
5     CONTINUE
      GO TO 98
4     DO 6 N=1,3
      GX=PBSOLN(LN,NODE(N),1) + GX
6     GY=GY + PBSOLN(LN,NODE(N),2)
      GX=(GX*2./9.+PBSOLE(LN,NEL,1)/3.)/2.
```

```
      GY=(GY*2./9.+PBSOLE(LN,NEL,2)/3.)/2.
      DO 8 N=1,3
      IF(NODE(N)-NJ) 9,9,8
 9    KPPRI(NODE(N),2*L-1)=KPPRI(NODE(N),2*L-1)+B(N)*GX
      KPPRI(NODE(N),2*L)=KPPRI(NODE(N),2*L)-A(N)*GY
 8    CONTINUE
 98   CONTINUE
C
C   ADD ELEMENT TEMPERATURE CURVATURE CONTRIBUTION TO ROTATION VECTOR
C
C   ADD THIS ROUTINE WHEN 'ELEMENT TEMPERATURE LOADS' COMMAND
C   BECOMES OPERATIONAL
C
 99   CONTINUE
      RELEASE KPPRI
      RELEASE ELEXT
      RELEASE ELTOP
      RELEASE JTXYZ
      RELEASE JINT
      RELEASE LEXT
      RELEASE IPRTIC
      RELEASE PBSOLE
      RELEASE PBSOLN
      RETURN
      END
```

```
      SUBROUTINE DICTS(IBN,IBC,NODE)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   FORTRAN IV
C
C     THIS IS A DICTIONARY SUBROUTINE FOR BRANCHING
C     TO THE APPROPRIATE SUBROUTINE FOR PROCESSING THE
C     STRETCHING BOUNDARY CONDITION
      GO TO (1,2,3,4,5,6,7,8),IBC
 1    CALL SDISPL(IBN,NODE)
      RETURN
 2    CALL SSTRES(IBN,NODE)
      RETURN
 3    CALL SELAST(IBN,NODE)
      RETURN
 4    CALL SEDGEB(IBN,NODE)
      RETURN
 5    CALL SMIXED(IBN,NODE)
      RETURN
 6    CALL SRIGID(IBN,NODE)
      RETURN
 7    CALL SRIGID(IBN,NODE)
      RETURN
 8    CALL SRIGID(IBN,NODE)
      RETURN
      END



      SUBROUTINE DICTB(IBN,IBC,NODE)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   FORTRAN IV
C
C     THIS IS A DICTIONARY SUBROUTINE TO BRANCH TO
C     THE APPROPRIATE SUBROUTINE FOR PROCESSING THE
C     BENDING BOUNDARY CONDITION
C
      GO TO (1,2,3,4,5,6),IBC
 1    CALL BDISPL(IBN,NODE)
      RETURN
 2    CALL BSTRES(IBN,NODE)
      RETURN
 3    CALL BELAST(IBN,NODE)
      RETURN
 4    CALL BEDGEB(IBN,NODE)
      RETURN
 5    CALL BMIX1D(IBN,NODE)
      RETURN
 6    CALL BMIX2D(IBN,NODE)
      RETURN
      END
```

```
      SUBROUTINE SDISPL(IBN,NODE)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE  INTRODUCES DISPLACEMENT BOUNDARY
C     CONDITIONS FOR THE STRETCHING PROBLEM ON BOUNDARY IBN
C     BEGINNING AT NODE POSITION -NODE-
C
      COMMON IMIN,FILL1(101),NLDSI,FILL2(63),KPPRI(P),FILL3(161)
      COMMON JINT(P),FILL4(86),BDCOND(P)
      DYNAMIC ARRAY JINT,BDCOND,KPPRI(R)
 4    I=BDCOND(IBN,NODE,1)
      NINT=JINT(I)
      IMIN=IMIN+1
      DO 1 L=1,NLDSI
      KPPRI(NINT,2*L-1)=BDCOND(IBN,NODE,4)
 1    KPPRI(NINT,2*L)=BDCOND(IBN,NODE,5)
      IF(BDCOND(IBN,NODE+1,1)) 2,2,3
 2    NODE=1
      GO TO 7
 3    NODE=NODE+1
 7    IF(BDCOND(IBN,NODE,2)-1) 4,4,5
 5    RELEASE BDCOND
      RELEASE JINT
      RELEASE KPPRI
      RETURN
      END
```

```
      SUBROUTINE SSTRES(IBN,NODE)
C
C    AUTHOR-     D. A. NAGY
C    LANGUAGE-   ICETRAN
C
C    THIS SUBROUTINE INTRODUCES STRESS BOUNDARY CONDITICNS
C    FOR THE STRETCHING PROBLEM ON BOUNDARY IBN BEGINNING
C    AT NODE  POSITION SPECIFIED BY -NODE-
C
      COMMON FILL1(102),NLDSI,FILL2(63),KPPRI(P),FILL3(102),JTXYZ(P)
      COMMON FILL4(57),JINT(P),FILL5(86),BDCOND(P)
      DYNAMIC ARRAY JTXYZ(R),BDCOND,KPPRI(R)
      N=1
      IF(NODE-1) 10,11,10
 11   IF(BDCOND(IBN,NODE+N,1)) 12,13,12
 13   NPR=BDCOND(IBN,NODE+N-1,1)
      GO TO 14
 12   N=N+1
      GO TO 11
 10   NPR=BDCOND(IBN,NODE-1,1)
 14   XP=JTXYZ(NPR,1)
      YP=JTXYZ(NPR,2)
      FXPR=0.
      FYPR=0.
      NC=BDCOND(IBN,NODE,1)
      XC=JTXYZ(NC,1)
      YC=JTXYZ(NC,2)
      FLP=SQRT((XC-XP)**2.+(YC-YP)**2.)
      FXC=BDCOND(IBN,NODE,4)
      FYC=BDCOND(IBN,NODE,5)
 20   NXT=BDCOND(IBN,NODE+1,1)
      IF(NXT) 1,1,2
 1    NEXT=1
      NXT=BDCOND(IBN,NEXT,1)
      GO TO 3
 2    NEXT=NODE+1
 3    XNX=JTXYZ(NXT,1)
      YNX=JTXYZ(NXT,2)
      FLC=SQRT((XNX-XC)**2.+(YNX-YC)**2.)
      INDIC=BDCOND(IBN,NEXT,2)
      GO TO (4,5,4,4,4,4,4,4),INDIC
 4    FXNX=0.
      FYNX=0.
      GO TO 6
 5    FXNX=BDCOND(IBN,NEXT,4)
      FYNX=BDCOND(IBN,NEXT,5)
 6    RXI=(FLP*(FXPR+2.*FXC)+FLC*(FXNX+2.*FXC))/6.
      RYI=(FLP*(FYPR+2.*FYC)+FLC*(FYNX+2.*FYC))/6.
      NINT=JINT(NC)
      DO 7 L=1,NLDSI
      KPPRI(NINT,2*L-1)=KPPRI(NINT,2*L-1)+RXI
 7    KPPRI(NINT,2*L)=KPPRI(NINT,2*L)+RYI
      NODE=NEXT
      GO TO (8,9,8,8,8,8,8,8),INDIC
 8    RELEASE BDCOND
```

```
      RELEASE JTXYZ
      RELEASE KPPRI
      RETURN
9     NPR=NC
      FXPR=FXC
      FYPR=FYC
      FLP=FLC
      NC=NXT
      XC=XNX
      YC=YNX
      FXC=FXNX
      FYC=FYNX
      GO TO 20
      END
```

```
      SUBROUTINE SMIXED(IBN,NODE)
C
C    AUTHOR-     D. A. NAGY
C    LANGUAGE-   ICETRAN
C
C
C    THIS SUBROUTINE PROCESSES THE MIXED BOUNDARY CONDITIONS
C    FOR THE PORTION OF BOUNDARY IBN BEGINNING WITH NODE.
C
      COMMON FILL1(100),NSOL,NDUM,NLDSI,FILL2(63),KPPRI(P),FILL3(102)
      COMMON JTXYZ(P)
      COMMON FILL5(45),NJ,FILL6(9),JINT(P),FILL7(54),NSYM,FILL8(31)
      COMMON BDCOND(P)
      DYNAMIC ARRAY KPPRI(R),JTXYZ(R),JINT,BDCOND
      DIMENSION A(4)
      N=1
C
C    DETERMINE NODE JUST BEFORE CURRENT NODE
C
      IF(NODE-1) 1,1,2
 1    IF(BDCOND(IBN,NODE+N,1)) 3,4,3
 4    NPR=BDCOND(IBN,NODE+N-1,1)
      GO TO 5
 3    N=N+1
      GO TO 1
 2    NPR=BDCOND(IBN,NODE-1,1)
C
C    DETERMINE COORDINATES OF PREVIOUS NODE, CURRENT NODE,
C    AND LENGTH OF SEGMENT CONNECTING THEM.
C
 5    XP=JTXYZ(NPR,1)
      YP=JTXYZ(NPR,2)
      NC=BDCOND(IBN,NODE,1)
      XC=JTXYZ(NC,1)
      YC=JTXYZ(NC,2)
      FLP=SQRT((XC-XP)**2.+(YC-YP)**2.)
C
C    DETERMINE NEXT NODE, ITS COORDINATES, AND LENGTH OF
C    SEGMENT FROM CURRENT NODE TO NEXT NODE.
C
 300  NX=BDCOND(IBN,NODE+1,1)
      IF(NX) 6,7,6
 7    NEXT=1
      NX=BDCOND(IBN,NEXT,1)
      GO TO 8
 6    NEXT=NODE+1
 8    XN=JTXYZ(NX,1)
      YN=JTXYZ(NX,2)
      FLC=SQRT((XN-XC)**2.+(YN-YC)**2.)
C
C    DETERMINE FORCE COMPONENT AND ANGLE.
C
      RQI=0.5*(FLC+FLP)*BDCOND(IBN,NODE,5)
      THETA=BDCOND(IBN,NODE,6)
      CT=COS(THETA)
```

```
      ST=SIN(THETA)
      URI=BDCOND(IBN,NODE,4)
      I=JINT(NC)
C
C   LOOP ON ALL ACTIVE NODES.
C
      DO 99 J=1,NJ
      IF(J-I) 9,10,9
C
C   MODIFY SUBMATRIX I,I OF GLOBAL STIFFNESS MATRIX AND
C   MODIFY SUBMATRIX I OF LOAD VECTOR KPPRI
C
 10   GO TO (50,51),NSYM
 50   CALL FNDSYM(N,I,I,A)
      GO TO 52
 51   CALL FNDNSM(N,I,I,A)
 52   DO 55 L=1,NLDSI
      SIX=KPPRI(I,2*L-1)
      SIY=KPPRI(I,2*L)
     0KPPRI(I,2*L)=RQI-ST*SIX+CT*SIY+URI*(ST*(CT*A(1)+ST*A(2)) -
     1  CT*(CT*A(3)+ST*A(4)))
 55   KPPRI(I,2*L-1)=URI
      A(4)=ST*ST*A(1)-ST*CT*(A(2)+A(3))+CT*CT*A(4)
      A(1)=1.
      A(2)=0.
      A(3)=0.
      GO TO (53,54),NSYM
 53   CALL STRSYM(I,I,A)
      GO TO 99
 54   CALL STRNSM(I,I,A)
      GO TO 99
C
C   MODIFY SUPER-ROW I OF GLOBAL STIFFNESS MATRIX (I=INTERNAL
C   NODE NUMBER OF CURRENT NODE), AND MODIFY SUBMATRICES 1
C   TO I-1 OF THE LOAD VECTOR KPPRI.
C
 9    GO TO (60,61),NSYM
 60   CALL FNDSYM(N,I,J,A)
      GO TO 62
 61   CALL FNDNSM(N,I,J,A)
 62   IF(N) 99,99,63
 63   IF(J-I) 64,65,65
 64   DO 66 L=1,NLDSI
      KPPRI(J,2*L-1)=KPPRI(J,2*L-1)-URI*(CT*A(1)+ST*A(3))
 66   KPPRI(J,2*L)=KPPRI(J,2*L)-URI*(CT*A(2)+ST*A(4))
 65   A(4)=-ST*A(2)+CT*A(4)
      A(3)=-ST*A(1)+CT*A(3)
      A(1)=0.
      A(2)=0.
      GO TO (67,68),NSYM
 67   CALL STRSYM(I,J,A)
      GO TO 99
 68   CALL STRNSM(I,J,A)
 99   CONTINUE
C
```

```
C   MODIFY SUPER-COLUMN I OF GLOBAL STIFFNESS MATRIX AND
C   SUBMATRICES I+1 TO  NSOL.
C
      DO 199 J=1,NJ
      IF(J-I) 20,199,20
 20   GO TO (21,22),NSYM
 21   CALL FNDSYM(N,J,I,A)
      GO TO 23
 22   CALL FNDNSM(N,J,I,A)
 23   IF(N) 199,199,24
 24   IF(J-I) 29,29,28
 28   IF(J-NSOL) 128,128,29
 128  DO 27 L=1,NLDSI
      KPPRI(J,2*L-U)=KPPRI(J,2*L-1)-URI*(CT*A(1)+ST*A(2))
 27   KPPRI(J,2*L)=KPPRI(J,2*L)-URI*(CT*A(3)+ST*A(4))
 29   A(4)=CT*A(4)-ST*A(3)
    A(2)=-ST*A(1)+CT*A(2)
      A(1)=0.
      A(3)=0.
      GO TO (25,26),NSYM
 25   CALL STRSYM(J,I,A)
      GO TO 199
 26   CALL STRNSM(J,I,A)
 199  CONTINUE
C
C   DETERMINE IF NEXT NODE ALONG BOUNDARY IS ALSO MIXED
C   BOUNDARY CONDITIONS TYPE.  IF SO,NEXT NODE BECOMES
C   CURRENT NODE, CURRENT NODE BECOMES PREVIOUS NODE,
C   AND ENTIRE PROCEDURE IS REPEATED FROM  STATEMENT 300.
C   IF NOT, SUBROUTINE RETURNS CONTROL TO STNBCM.
C
      NCOND=BDCOND(IBN,NEXT,2)
      IF(NCOND-5) 200,201,200
 200  RELEASE BDCOND
      RELEASE JTXYZ
      RELEASE JINT
      RELEASE KPPRI
      NODE=NEXT
      RETURN
 201  NPR=NC
      XP=XC
      YP=YC
      FLP=FLC
      NC=NX
      XC=XN
      YC=YN
      NODE=NEXT
      GO TO 300
      END
```

```
      SUBROUTINE FNDSYM(N,I,J,A)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE RETRIEVES 2X2 SUBMATRIX I,J FROM THE SYMMETRIC
C     GLOBAL STIFFNESS/FLEXIBILITY MATRIX AND STORES IT IN THE ONE-
C     DIMANSIONAL ARRAY A.
C     IF THE SUBMATRIX IS ZERO OR ABOVE THE DIAGONAL, THE VALUE OF N IS
C     RETURNED =0
C
      COMMON FILL1(160),KDIAG(P),KOFDG(P),IOFDG(P)
      DYNAMIC ARRAY KDIAG(R),KOFDG(R),IOFDG
      DIMENSION A(4)
      IF(I-J) 1,2,3
 2    DO 4 L=1,4
 4    A(L)=KDIAG(I,L)
      N=2
      RETURN
 1    N=1
      RETURN
 3    M=IOFDG(I,1)
      DO 7 K=1,M
      L=IOFDG(I,K*2)
      IF(L-J) 7,8,7
 7    CONTINUE
      N=1
      RETURN
 8    MPOS=IOFDG(I,K*2+1)
      DO 9 L=1,4
 9    A(L)=KOFDG(MPOS,L)
      N=2
      RETURN
      END
```

```
      SUBROUTINE STRSYM(I,J,A)
C
C    AUTHOR-     D. A. NAGY
C    LANGUAGE-   ICETRAN
C
C    THIS SUBROUTINE TRANSFERS SUBMATRIX I,J FROM ARRAY A BACK INTO
C    THE SYMMETRIC GLOBAL STIFFNESS/FLEXIBILITY MATRIX.
C
      COMMON FILL1(160),KDIAG(P),KOFDG(P),IOFDG(P)
      DYNAMIC ARRAY KDIAG(R),KOFDG(R),IOFDG
      DIMENSION A(4)
      IF(I-J) 1,2,3
 1    RETURN
 2    DO 4 L=1,4
 4    KDIAG(I,L)=A(L)
      RETURN
 3    M=IOFDG(I,1)
      DO 5 L=1,M
      IF(IOFDG(I,L*2)-J) 5,6,5
 5    CONTINUE
      RETURN
 6    MPOS=IOFDG(I,L*2+1)
      DO 7 L=1,4
 7    KOFDG(MPOS,L)=A(L)
      RETURN
      END


      SUBROUTINE FNDNSM(N,I,J,A)
C
C    AUTHOR-     D. A. NAGY
C    LANGUAGE-   ICETRAN
C
C    THIS SUBROUTINE RETRIEVES 2X2 SUBMATRIX I,J FROM THE NON-SYMMETRIC
C    GLOBAL STIFFNESS/FLEXIBILITY MATRIX AND STORES IT IN THE ONE-
C    DIMENSIONAL ARRAY A.  IF THE SUBMATRIX IS ZERO, THE VALUE OF N
C    IS RETURNED =0
C
      COMMON FILL1(168),FCMAT(P),DUM(2),IREL1(P)
      DYNAMIC ARRAY IREL1,FCMAT
      DIMENSION A(4)
      N=IREL1(2*I,J)+1
      GO TO (1,2),N
 1    RETURN
 2    A(1)=FCMAT(J,2*I-1,1)
      A(2)=FCMAT(J,2*I-1,2)
      A(3)=FCMAT(J,2*I,1)
      A(4)=FCMAT(J,2*I,2)
      RETURN
      END
```

```
      SUBROUTINE STRNSM(I,J,A)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE TRANSFERS THE SUBMATRIX I,J FROM ARRAY A BACK
C     INTO THE NON-SYMMETRIC STIFFNESS/FLEXIBILITY MATRIX.
C
      COMMON FILL1(168),FCMAT(P),ICUREL(P),IREL1(P)
      DYNAMIC ARRAY FCMAT,ICUREL,IREL1
      DIMENSION A(4)
      II=IDEF(JNK,JNK,JNK,FCMAT,J,2*I)
      IF(II) 1,2,1
 2    DEFINE FCMAT(J,2*I-1),2,FULL
      DEFINE FCMAT(J,2*I),2,FULL
      DESTROY FCMAT
      IREL1(2*I-1,J)=1
      IREL1(2*I,J)=1
      N=IDEF(JNK,JNK,JNK,ICUREL,J)
      DO 3 L=1,N
      IF(ICUREL(J,L)-2*I-1) 3,4,4
 3    CONTINUE
      GO TO 6
 4    DEFINE ICUREL(J),N+2,HALF
      KK=N-L+1
      DO 5 K=1,KK
 5    ICUREL(J,N+3-K)=ICUREL(J,N+1-K)
      ICUREL(J,L)=2*I-1
      ICUREL(J,L+1)=2*I
 1    FCMAT(J,2*I-1,1)=A(1)
      FCMAT(J,2*I-1,2)=A(2)
      FCMAT(J,2*I,1)=A(3)
      FCMAT(J,2*I,2)=A(4)
      RETURN
 6    DEFINE ICUREL(J),N+2,HALF
      ICUREL(J,N+1)=2*I-1
      ICUREL(J,N+2)=2*I
      GO TO 1
      END
```

A7-6 NON-SYMMETRIC SOLVER INTERFACE PROGRAM.

```
      SUBROUTINE STNSSL
C
C   AUTHOR-     D. A. NAGY
C   LANGUAGE-   ICETRAN
C
C   THIS SUBROUTINE ACTS AS AN INTERFACE BETWEEN THE SYSTEM
C   AND A NON-SYMMETRIC SOLVER ORIGINALLY WRITTEN FOR ANOTHER
C   SYSTEM.  IT TRANSFERS THE PORTION OF THE LOAD/ROTATION
C   VECTOR FOR EACH INDEPENDENT LOADING TO AN ARRAY COMPATIBLE
C   WITH THE SOLVER AND THEN CALLS THE SOLVER ONCE FOR EACH
C   INDEPENDENT LOADING CONDITION.  AFTER EACH SOLUTION, IT
C   TRANSFERS THE RESULTS BACK INTO THE ARRAY KPPRI FOR USE
C   BY THE BACKSUBSTITUTION PHASE PROGRAMS.
C
      COMMON KDANG,NCUT,FILL1(98),NSOL,NN,NLDSI,FILL2(59),FPMAT(P)
      COMMON FCFOR(P),KPPRI(P)
      DYNAMIC ARRAY FPMAT,FCFOR,KPPRI(R)
      NCUT=NSOL
      DEFINE FPMAT,2*NCUT,FULL
C
C   LOOP ON ALL ACTIVE, INDEPENDENT LOADING CONDITIONS.
C   TRANSFER LOAD/ROTATIONS VECTOR PORTION FROM KPPRI TO
C   FPMAT AND CALL NON-SYMMETRIC SOLVER STNDUM
C
      DO 99 L=1,NLDSI
      DO 98 N=1,NCUT
      FPMAT(2*N-1)=KPPRI(N,2*L-1)
 98   FPMAT(2*N)=KPPRI(N,2*L)
      CALL STNDUM
C
C   TRANSFER RESULTS FROM FCFOR BACK TO KPPRI
C
      DO 97 N=1,NCUT
      KPPRI(N,2*L-1)=FCFOR(N,1)
 97   KPPRI(N,2*L)=FCFOR(N,2)
 99   CONTINUE
      NSOL=NCUT
      RELEASE KPPRI
      DESTROY FPMAT
      DESTROY FCFOR
      RETURN
      END
```

A7-7 BACKSUBSTITUTION PHASE.

```
      SUBROUTINE STNBKS
C
C   AUTHOR-     D. A. NAGY
C   LANGUAGE-   ICETRAN
C
C   THIS IS THE EXECUTIVE FOR THE BACKSUBSTITUTION (AND
C   TEMPORARILY THE OUTPUT) PHASE.
C
      ODIMENSION ND(3),X(3),Y(3),U(3),V(3),A(3),B(3),E(3),S(3),PS(2),
     1 PE(2)
       COMMON FILL1(100),NSOL,DFILL(49),LEXT(P),FILL2(14),KPPRI(P)
       COMMON FILL3(87),LEXTN
       COMMON LTYP(P),FILL4(3),LDTLE(P),FILL5(3),JEXTN,FILL6(5),JTXYZ(P)
       COMMON FILL7(47),NJ,NFILL,NLDSI,FILL8(9),JEXT(P),FILL9(30)
       COMMON ELPROP(P),FILL10(18),NBXTEL,NDFIL,NSYM,FILL11(12),NODISP(P
       COMMON STRAIN(P)
       COMMON STRESS(P),PRSTRN(P),PRSTRS(P),FILL12(5),IPROB,FILL13(11)
       COMMON SFTEMP(P),RNDTEM(P)
       DYNAMIC ARRAY NODISP(R),SFTEMP,STRESS,STRAIN,PRSTRS,PRSTRN
       DYNAMIC ARRAY RNDTEM,JEXT,LDTLE(R),ELTOP(I),JTXYZ(R),ELPROP
       DYNAMIC ARRAY KPPRI(R),LEXT
       DYNAMIC ARRAY LTYP
C
C   DEFINE NODISP AND TRANSFER RESULTS FROM KPPRI TO NODISP.
C   COMPUTE RESULTS FOR DEPENDENT LOADING CONDITIONS.
C
       DEFINE NODISP,LEXTN,JEXTN,6
       DESTROY NODISP
       DO 60 I=1,NJ
       DO 60 L=1,NLDSI
       LDN=LEXT(L)
       JTN=JEXT(I)
       NODISP(LDN,JTN,1)=KPPRI(I,2*L-1)
 60    NODISP(LDN,JTN,2)=KPPRI(I,2*L)
       DO 61 L=1,LEXTN
       J=LTYP(L)+1
       GO TO (62,62,61,61,62,62,61,61),J
 62    NLC=LDTLE(L,17)
       DO 63 M=1,NLC
       DO 63 N=1,JEXTN
      ONODISP(L,N,1)=NODISP(L,N,1)+NODISP(LDTLE(L,16+2*M),N,1)*LDTLE(L,
     1 16+2*M+1)
 63   ONODISP(L,N,2)=NODISP(L,N,2)+NODISP(LDTLE(L,16+2*M),N,2)*LDTLE(L,
     1 17+2*M)
 61    CONTINUE
       IF(IPROB) 1,2,2
 1     IF(IPROB+2) 3,3,2
C
C   DEFINE ARRAYS FOR STORAGE OF ELEMENT STRESSES AND
C   STRAINS, PRINCIPAL STRESSES AND PRINCIPAL STRAINS.
C
 2     DEFINE STRESS,LEXTN,NBXTEL,6
```

```
      DESTROY STRESS
      DEFINE STRAIN,LEXTN,NBXTEL,6
      DESTROY STRAIN
      DEFINE PRSTRS,LEXTN,NBXTEL,4
      DESTROY PRSTRS
      DEFINE PRSTRN,LEXTN,NBXTEL,4
      DESTROY PRSTRN
C
C   LOOP ON ALL ELEMENTS
C
 3    DO 99 NE=1,NBXTEL
      DO 4 I=1,3
      ND(I)=ELTOP(NE,I+1)
C
C   OBTAIN COORDINATES OF NODES INCIDENT UPON CURRENT ELEMENT
C   AND ELEMENT PROPERTIES.
C
      X(I)=JTXYZ(ND(I),1)
 4    Y(I)=JTXYZ(ND(I),2)
      A(1)=-X(2)+X(3)
      A(2)=-X(3)+X(1)
      A(3)=-X(1)+X(2)
      B(1)=-Y(2)+Y(3)
      B(2)=-Y(3)+Y(1)
      B(3)=-Y(1)+Y(2)
      AREA=0.5*ABS(X(1)*B(1)+X(2)*B(2)+X(3)*B(3))
      CTX=ELPROP(NE,10)
      IF(CTX) 5,6,5
 6    CTX=ELSTCT
 5    CTY=ELPROP(NE,11)
      IF(CTY) 7,8,7
 8    CTY=CTX
 7    EX=ELPROP(NE,6)
      IF(EX) 9,10,9
 10   EX=ELSTDE
 9    EY=ELPROP(NE,7)
      IF(EY) 11,12,11
 12   EY=EX
 11   PX=ELPROP(NE,8)
      IF(PX) 13,14,13
 14   PX=ELSTPO
 13   G=ELPROP(NE,12)
      IF(G) 15,16,15
 16   G=ELSTDG
 15   H=ELPROP(NE,2)
      DO 99 LO=1,LEXTN
      DO 17 I=1,3
      U(I)=NODISP(LO,ND(I),1)
 17   V(I)=NODISP(LO,ND(I),2)
C
C   CALL STESTR OR STESCP TO COMPUTE ELEMENT STRESSES,
C   STRAINS,ETC., DEPENDING ON WHETHER THE PROBLEM JUST
C   SOLVED WAS STRETCHING OR BENDING.
C
      IF(IPROB) 18,19,19
```

```
 18   IDEL=3
      JDEL=2
      CALL STESCP(EX,EY,PX,CTX,CTY,G,H,LO,NE,AREA,A,B,U,V,E,S,PS,PE)
      GO TO 20
 19   IDEL=0
      JDEL=0
      CALL STESTR(EX,EY,PX,CTX,CTY,G,LO,NE,AREA,A,B,U,V,E,S,PS,PE)
 20   DO 21 I=1,2
      STRESS(LO,NE,I+IDEL)=S(I)
      STRAIN(LO,NE,I+IDEL)=E(I)
      PRSTRS(LO,NE,I+JDEL)=PS(I)
 21   PRSTRN(LO,NE,I+JDEL)=PE(I)
      STRESS(LO,NE,3+IDEL)=S(3)
 99   STRAIN(LO,NE,3+IDEL)=E(3)
      IT=IPROB+3
      GO TO (22,22,24,23,23),IT
C
C    STORE STRESS FUNCTIONS IN ARRAY SFTEMP AND TRANSFER
C    NODAL DISPLACEMENTS (X AND Y COMPONENTS) FROM RNDTEM
C    BACK TO NODISP.
C
 22   DEFINE SFTEMP,LEXTN,JEXTN,2
      DO 25 I=1,LEXTN
      DO 25 J=1,JEXTN
      SFTEMP(I,J,1)=NODISP(I,J,1)
      SFTEMP(I,J,2)=NODISP(I,J,2)
      GO TO (26,27),IT
 26   NODISP(I,J,1)=RNDTEM(I,J,1)
      NODISP(I,J,2)=RNDTEM(I,J,2)
      GO TO 25
 27   NODISP(I,J,1)=0.
      NODISP(I,J,2)=0.
 25   CONTINUE
      GO TO (28,29),IT
C
C    TRANSFER NODAL DISPLACEMENTS TO TEMPORARY ARRAY RNDTEM
C    SO THAT NODISP WILL BE FREE FOR BENDING SOLUTION PHASE,
C    WHICH IS NEXT.
C
 23   DEFINE RNDTEM,LEXTN,JEXTN,2
      DO 30 I=1,LEXTN
      DO 30 J=1,JEXTN
      RNDTEM(I,J,1)=NODISP(I,J,1)
 30   RNDTEM(I,J,2)=NODISP(I,J,2)
      RELEASE RNDTEM
      GO TO 24
 29   DESTROY RNDTEM
 28   RELEASE SFTEMP
 24   RELEASE STRESS
      RELEASE STRAIN
      RELEASE PRSTRS
      RELEASE PRSTRN
      RELEASE NODISP
      RELEASE ELTOP
      RELEASE JTXYZ
```

```
      RELEASE ELPROP
      DESTROY KPPRI
      IF(IPROB-2) 31,32,32
C
C     IF PROBLEM IS COMPLETELY FINISHED, CALL TEMPORARY OUTPUT
C     SUBROUTINE STNOUT AND RETURN.
C
 31   CALL STNOUT
      RETURN
C
C     IF PROBLEM IS GENERAL PLATE PROBLEM, CONSTRUCT STACK
C     OF PROGRAMS FOR BENDING SOLUTION PHASE AND TRANSFER
C     TO THE FIRST PROGRAM IN THE STACK.
C
 32   IPROB=-2
      NSOL=NJ
      IF(NSYM-1) 31,100,101
 100  ADD TO STACK (1,'STNBKS')
      ADD TO STACK (1,'STSLVR')
      ADD TO STACK (1,'STNBCM')
      ADD TO STACK (1,'STEASS')
 102  ADD TO STACK (1,'STNGEN')
      TRANSFER TO STACK
 101  ADD TO STACK (1,'STNSAS')
      GO TO 102
      END
```

```
      SUBROUTINE STESTR(EX,EY,PX,CX,CY,G,L,N,AR,A,B,U,V,E,S,PS,PE)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE COMPUTES THE ELEMENT STRESSES,STRAINS,
C     PRINCIPAL STRESSES, AND PRINCIPAL STRAINS FOR THE ELEMENT
C     WHOSE PROPERTIES ARE PASSED TO IT AS ARGUMENTS.
C     THE COMPUTED VALUES ARE RETURNED IN THE DIMENSIONED
C     ARRAYS E,S,PS, AND PE.
      DIMENSION A(3),B(3),U(3),V(3),E(3),S(3),PE(2),PS(2)
      COMMON FILL(367),ELOADS(P)
      DYNAMIC ARRAY ELOADS
      E(1)=-0.5*(B(1)*U(1)+B(2)*U(2)+B(3)*U(3))/AR
      E(2)=0.5*(A(1)*V(1)+A(2)*V(2)+A(3)*V(3))/AR
     0E(3)=0.5*(A(1)*U(1)-B(1)*V(1)+A(2)*U(2)-B(2)*V(2)+A(3)*U(3)-
     1 B(3)*V(3))/AR
      II=IDEF(JNK,JNK,JNK,ELOADS,N,L)
      IF(II) 1,1,2
C
C        ADD TEMPERATURE STRAIN ROUTINE HERE LATER
C        DUMMY STATEMENT NO. 2 FOR NOW . . .
C
 2    GO TO 1
 1    EXT=0.
      EYT=0.
 3    EXL=E(1)-EXT
      EYL=E(2)-EYT
      PY=EX*PX/EY
      S(1)=EX*(EXL+PX*EYL)/(1.-PX*PY)
      S(2)=EY*(EYL+PY*EXL)/(1.-PX*PY)
      S(3)=G*E(3)
      SXP=SQRT(((S(1)-S(2))**2.)/2. + S(3)**2.)
      SYP=(S(1)+S(2))/2.
      PS(1)=SXP+SYP
      PS(2)=SYP-SXP
      EXP=SQRT(((E(1)-E(2))**2.)/2. + (E(3)/2.)**2.)
      EYP=(E(1)+E(2))/2.
      PE(1)=EXP+EYP
      PE(2)=EYP-EXP
      RETURN
      END
```

```
      SUBROUTINE STESCP(EX,EY,PX,CX,CY,G,H,L,N,AR,A,B,U,V,E,S,PS,PE)
C
C     AUTHOR-     D. A. NAGY
C     LANGUAGE-   ICETRAN
C
C     THIS SUBROUTINE COMPUTES THE BENDING STRESS COUPLES,
C     CURVATURES, PRINCIPAL COUPLES, AND PRINCIPAL CURVATURES
C     FOR THE ELEMENT WHOSE PROPERTIES ARE PASSED TO IT AS
C     ARGUMENTS.  THE RESULTS ARE RETURNED IN THE DIMENSIONED
C     ARRAYS E,S,PS,AND PE.
C
      DIMENSION A(3),B(3),U(3),V(3),E(3),S(3),PE(2),PS(2),NODE(3)
      COMMON FILL1(365),ELTOP(P),ELOADS(P),FILL2(52),PBSOLN(P)
      COMMON PBSOLE(P),FILL3(4),IPRTIC(P)
      DYNAMIC ARRAY ELOADS,PBSOLE,IPRTIC,PBSOLN,ELTOP(I)
      PY=EX*PX/EY
      DX=EX*(H**3.)/(12.*(1.-PX*PY))
      DY=EY*DX/EX
      IF(IPRTIC(L)) 1,1,2
 2    RKX=PBSOLE(L,N,1)/AR
      RKY=PBSOLE(L,N,2)/AR
      GO TO 8
 1    IF(PBSOLE(L,N,1)) 3,4,3
 4    IF(PBSOLE(L,N,2)) 3,5,3
 3    RKX=PBSOLE(L,N,1)
      RKY=PBSOLE(L,N,2)
      GO TO 8
 5    DO 6 I=1,3
 6    NODE(I)=ELTOP(N,I+1)
      RKX=0.
      RKY=0.
      DO 7 I=1,3
      RKX=RKX+PBSOLN(L,NODE(I),1)/3.
 7    RKY=RKY+PBSOLN(L,NODE(I),2)/3.
 8    SCXP=-DX*(RKY+PX*RKX)
      SCYP=-DY*(RKX+PY*RKY)
      S(1)=0.5*(A(1)*V(1)+A(2)*V(2)+A(3)*V(3))/AR
      S(2)=-0.5*(B(1)*U(1)+B(2)*U(2)+B(3)*U(3))/AR
     0S(3)=-0.25*(A(1)*U(1)-B(1)*V(1)+A(2)*U(2)-B(2)*V(2)+A(3)*U(3) -
     1  B(3)*V(3))/AR
      E(1)=12.*(S(1)-PY*S(2))/(EX*H**3.)-RKY
      E(2)=12.*(S(2)-PX*S(1))/(EY*H**3.)-RKX
      E(3)=(12./(G*H**3.))*S(3)
      II=IDEF(JNK,JNK,JNK,ELOADS,N,L)
      IF(II) 9,9,10
C
C        ADD TEMPERATURE CURVATURE ROUTINE HERE LATER
C        DUMMY STATEMENT NO. 10 FOR NOW
C
 10   GO TO 9
 9    EXT=0.
      EYT=0.
 11   E(1)=E(1)+EXT
      E(2)=E(2)+EYT
      S(1)=S(1)+SCXP
```

```
S(2)=S(2)+SCYP
SXP=SQRT(((S(1)-S(2))**2.)/2. + S(3)**2.)
SYP=(S(1)+S(2))/2.
PS(1)=SXP+SYP
PS(2)=SYP-SXP
EXP=SQRT(((E(1)-E(2))**2.)/2. + E(3)**2.)
EYP=(E(1)+E(2))/2.
PE(1)=EXP+EYP
PE(2)=EYP-EXP
RELEASE PBSOLE
RELEASE PBSOLN
RELEASE ELTOP
RELEASE IPRTIC
RETURN
END
```

```
      SUBROUTINE STNOUT
C
C     AUTHOR-  D. A. NAGY
C
C     LANGUAGE-  ICETRAN
C
C     THIS SUBROUTINE IS A TEMPORARY OUTPUT SUBROUTINE
C     INCLUDED IN THE LOAD MODULE STNBKS.  IT OUTPUTS THE
C     NODAL DISPLACEMENTS FOR THE STRETCHING PROBLEM ONLY, THE
C     ELEMENT CENTERED VALUES OF STRESS,STRAIN,PRINCIPAL
C     STRESS, AND PRINCIPAL STRAIN FOR THE BENDING AND STRETCHING
C     PROBLEMS.  THIS OUTPUT IS PRODUCED WITHOUT THE REQUEST
C     OF THE USER.
C
      COMMON FILL1(250),LDID(P),FILL2,LEXTN,FILL3(7),JTID(P),FILL4,JEXTN
      COMMON FILL5(94),ELID(P),FILL6(22),NBXTEL,FILL7(14),NODISP(P)
      COMMON STRAIN(P),STRESS(P),PRSTRN(P),PRSTRS(P)
      DYNAMIC ARRAY LDID(D),JTID(D),ELID(D),NODISP(R),STRESS,STRAIN
      DYNAMIC ARRAY PRSTRS,PRSTRN
      DIMENSION DATA(6),PDATA(4)
      DOUBLE PRECISION DUM1
      WRITE (6,101)
      WRITE (6,117)
      DO 99 L=1,LEXTN
      DUM1=LDID(L)
      WRITE (6,101)
      WRITE (6,100) DUM1
      WRITE (6,101)
      WRITE (6,102)
      WRITE (6,103)
      DO 10 J=1,JEXTN
      DUM1=JTID(J)
      DUM2=NODISP(L,J,1)
      DUM3=NODISP(L,J,2)
   10 WRITE (6,104) DUM1,DUM2,DUM3
      WRITE (6,101)
      WRITE (6,105)
      WRITE (6,106)
      DO 20 I=1,NBXTEL
      DUM1=ELID(I)
      DO 21 N=1,6
   21 DATA(N)=STRESS(L,I,N)
   20 WRITE (6,107) DUM1, DATA
      WRITE (6,101)
      WRITE (6,108)
      WRITE (6,109)
      DO 30 I=1,NBXTEL
      DUM1=ELID(I)
      DO 31 N=1,6
   31 DATA(N)=STRAIN(L,I,N)
   30 WRITE (6,110) DUM1, DATA
      WRITE (6,101)
      WRITE (6,111)
      WRITE (6,112)
      DO 40 I=1,NBXTEL
```

```
      DUM1=ELID(I)
      DO 41 N=1,4
41    PDATA(N)=PRSTRS(L,I,N)
40    WRITE (6,113) DUM1, PDATA
      WRITE (6,101)
      WRITE (6,114)
      WRITE (6,115)
      DO 50 I=1,NBXTEL
      DUM1=ELID(I)
      DO 51 N=1,4
51    PDATA(N)=PRSTRN(L,I,N)
50    WRITE (6,116) DUM1, PDATA
99    CONTINUE
      RETURN
100   FORMAT(19H LOADING CONDITION , A8)
101   FORMAT(1H )
102   FORMAT(20H NODAL DISPLACEMENTS)
103   FORMAT(23H  NODE       U          V)
104   FORMAT(A8,2F10.6)
105   FORMAT(17H ELEMENT STRESSES)
106  0FORMAT(65H ELEMENT     SX         SY         SZ         MX         MY
     1       MXY)
107   FORMAT(A8,3F10.2,3F10.1)
108   FORMAT(16H ELEMENT STRAINS)
109  0FORMAT(66H ELEMENT     EX         EY      GAMMA-XY     CHI-X      CHI-Y
     1     CHI-XY)
110   FORMAT(A8,3F10.7,3F10.6)
111   FORMAT(27H ELEMENT PRINCIPAL STRESSES)
112   FORMAT(44H ELEMENT     S1         S2         M1         M2)
113   FORMAT(A8,2F10.2,2F10.1)
114   FORMAT(26H ELEMENT PRINCIPAL STRAINS)
115   FORMAT(45H ELEMENT     E1         E2       CHI-1      CHI-2)
116   FORMAT(A8,2F10.7,2F10.6)
117   FORMAT(38H UNITS ARE INCHES, POUNDS, AND RADIANS)
      END
```

APPENDIX 8

Programmer's Information

A8-1 Introduction

This appendix is intended to serve as a guide to someone with a good knowledge of FORTRAN-IV programming for completing the debugging of the system developed in this thesis or adding to its capabilities.

A8-2 Programming Languages

a) All computer programming for the system developed in this thesis was done within the context of the Integrated Civil Engineering System (ICES) being developed by the MIT Department of Civil Engineering. For a general overview of ICES, the programmer should consult

"ICES: CONCEPTS AND FACILITIES," Department of Civil Engineering, MIT, 1965.

b) The routines for reception of input,i.e., the definitions of the problem-oriented commands described in Chapter 4, were written in Command Definition Language (CDL), which is itself a problem-oriented language that may be learned relatively quickly. For a discussion of the philosophy of problem-oriented languages within the context of ICES, refer to Chapter of "ICES: Concepts and Facilities." The User's Manual for CDL is Chapter 3 of

"ICES: Programmer's Guide," Department of Civil Engineering, MIT, 1965.

and provides a complete description and some examples of CDL commands. If additional questions arise concerning some aspect of CDL, the programmer should consult Mr. Ronald A. Walter, instructor in the MIT Department of Civil Engineering, who wrote the Command Definition Language.

c) The subroutines for processing and storage of input data, solution of the specified problem(s), and output of

results were written in ICETRAN (ICES-FORTRAN). ICETRAN is the FORTRAN-IV language with the added capability of more flexible data array storage (dynamic memory allocation) and other forms of program linkage in addition to the CALL statement of FORTRAN. ICETRAN programs are first processed by a pre-compiler that translates the program into legitimate FORTRAN with calls to ICES library subroutines to perform the various tasks of the ICETRAN statements. The resulting translated program is then processed by the FORTRAN-IV compiler.

The philosophy of dynamic memory allocation and dynamic program linkage is discussed in Chapt. of "ICES: Concepts and Facilities." The detailed description of all ICETRAN commands is contained in Chapter 2 of "ICES: Programmer's Guide."

A8-3 STRUDL Subsystem and the Finite Element Analyzer

The programming system of this thesis was developed in the form of additions to STRUDL (STRUctural Design Language), a problem-oriented language subsystem of ICES for problems in structural engineering. In particular, these additions apply directly to the Finite Element Analyzer, a portion of the STRUDL subsystem which deals directly with the application of the finite element method. Initial documentation of the Finite Element Analyzer is given in

Ferrante,A.J., "A System for Finite Analysis," S.M. Thesis, Department of Civil Engineering, MIT, January 1967.

but its author Mr. Ferrante, an instructor in the MIT Civil Engineering Department, should be contacted to obtain the most recent version. The programmer should be thoroughly familiar with the input commands, dynamic ICETRAN data arrays, and COMMON data storage of the Finite Element Analyzer before working with the programs of this thesis. The data arrays used to store the symmetric global stiffness/flexibility matrix are most completely described in

MEMO IS 21.3 - Non Linear Analysis in STRUDL - Data Structure.

The arrays used for the storage of the non-symmetric global matrix are given here for lack of current documentation elsewhere.

1. Storage array:

   FCMAT(I,J,K)

   where

   I= hypercolumn of the global structural stiffness/ flexibility matrix

   J= row

   K= column within hypercolumn I (1 or 2)

2. Bookkeeping arrays:

   ICUREL(I,J)

   where

   I= hypercolumn

   J= number of Jth non-zero row in Ith hypercolumn

   - - -

   IRELl(I,J)

   where

   I= row of the global matrix

   J= 0 if Jth hypercolumn of row I is zero

   = 1 if Jth hypercolumn of row I is non-zero

A8-4 Running of CDL, ICETRAN, and STRUDL-Finite Element Analyzer Programs on the Computer

The programs of this thesis were written to be run under the CESL (Civil Engineering Systems Laboratory) Monitor on the IBM System/360 computer in the MIT Department of Civil Engineering. A description of the necessary control cards and program deck arrangements for all types of computer runs necessary to modify and use the system of this thesis is given in

> IG MEMO 32, "The Use of CESL ICES and OS ICES to Create, Modify, and Run Subsystems," MIT Department of Civil Engineering, April 26, 1967.

Decks to be run are submitted in the bins located in room 1-147,

adjacent to the IBM System/360 room. Jobs with an estimated running time of less than five minutes are submitted in the express bin and are "usually" run twice a day. Jobs of longer duration are run during the night if time permits.

A8-5 Interpretation of Output and Debugging Aids

a) CDL - execution of CDL programs includes reprinting each CDL command. Any error messages immediately follow the command that caused them, and are usually quite adequate for understanding the error committed. If the number 1 appears on the line following the FILE command ending a CDL program, it means that the program was interpreted successfully and the command defined is now a part of the complete dictionary of commands on the disk files.

b) ICETRAN SUBROUTINES - compilation of ICETRAN subroutines begins with a complete listing of the ICETRAN program as it is being processed by the precompiler. Any error messages immediately follow the statement that caused them. A complete description of the meaning of ICETRAN Precompiler error messages is contained in Chapt. of "ICES: Programmer's Guide." Following the listing of each subroutine is a COMMON map giving the relative location of each COMMON variable from the beginning of COMMON.

If no errors are detected by the precompiler, the programs are then processed by the FORTRAN compiler. In this process, the translated FORTRAN program is listed, with each line numbered consecutively. This listing is followed by a list of COMMON variables, local variables, constants, and statement numbers, with the relative locations of each given in the hexadecimal (base 16) number system. If any errors were detected in the Fortran compilation, the messages are printed after the list of COMMON variables, local variables, constants, and statement numbers. A complete explanation of these error messages may be found in

IBM System/360 BPS, FORTRAN IV 360P-FO-031, Programmer's Guide, C28-6583 (Appendix A).

A discussion of some of the more common error messages in compilation and execution of FORTRAN programs, and bugs in the FORTRAN compiler, may be found in

CESL MEMO 25 (Mrs. Jane Jordan),"User's Guide to CESL Monitor System," May 5, 1967.

Any problems not clarified by the above two documents should be taken to Mrs. Jane Jordan in Room 1-153.

c)Load Module Generation - the grouping of subroutines together in a package and storage of this package on the disk files of the STRUDL Subsystem is accomplished by Linkage Editor runs. These runs precompile and compile each subroutine, as discussed in item b) above. If no errors are detected in any of the compilations, the package of programs is filed on the disk. A storage map is printed giving the relative locations (in hexadecimal number system) of all subroutines and ICES library routines included in the Load Module. If all these routines are found during the packaging and added to the load module package, the message LOADING COMPLETE is printed.

d)Subsystem Execution - if errors occur during execution of one of the subroutines of the subsystem, the error message causing termination of the job may not always make the nature of the error evident. In the case of any error in dynamic array storage or retrieval, a fairly brief error message is printed giving the nature of the error and hexadecimal program location (object program) where the error occurred. By referring to the list of relative locations of statement numbers at the end of the Fortran compilation and the Storage Map at the end of the Load Module formation run, the programmer can determine approximately where the error occurred in the original ICETRAN source program.

Example:

The following message might appear during subsystem execution:

1014479 4000E48C
****
ERROR IN DYNAMIC ARRAY STORAGE OR RETRIEVAL AT THE PROGRAM LOCATION SHOWN ABOVE

The last four digits of the number undelined by asterisks is the program location. From this number should be subtracted the number DF00, which is the location of the first subroutine of the load module currently being used.

```
 E48C
-DF00
  58C
```

This number is then the relative location of the error within the load module. Referring to the Storage Map of the load module, it is seen that the first subroutine of the load module is stored at 2600. Thus adding 58C to 2600 will yield the relative location where the error occurred in the Storage Map. It may then easily be seen in which program this location falls. Subtracting the storage map location of the beginning of the particular program from 2600+58C will then yield the relative location within the program. This location may be compared to the relative locations of numbered statements, which is found at the end of the FORTRAN compilation of the program.

Errors other than dynamic array storage and retrieval are more difficult to trace. The following debugging commands may be inserted in the subsystem job runs to obtain more information about the execution of the job at the time that the error occurred:

DBGXCO

This command causes the course of action of the Execution Coordinator to be explicitly stated from the point where DBGXCO is inserted until the end of the job. The Execution Coordinator obtains modules from the disk files and directs the computer to execute particular programs according to the specifications of CDL or the previous

subroutine executed. The programmer can then tell if all programs that are supposed to be executed actually are being executed in the proper order and also exactly which program was being executed when the error occurred.

DBGALL

This command causes a complete dump of the core storage area where the current load module is stored during its execution. The dump occurs after the error is detected, so that it shows the status of the program at the time that the error occured. Mrs. Jane Jordan should be consulted for interpretation of the results of the dump.

POOLDP

This command causes a complete dump of COMMON storage area plus the data pool of arrays currently in the computer.

PRINT DATA

This command causes all data relating to nodes that has been input by the user to be printed out.

NAGY DEBUG

This command prints out the constructed boundary chains, specified boundary conditions, dislocations, nodal loadings, and specified values of the particular bending solution or computed exact values.

For additional debugging aids, interpretation of debugging information, and general counseling on location of bugs in the system, the programmer is referred to Mrs. Jane Jordan in Room 1-153.

APPENDIX 9

SAMPLE PROBLEM

```
/JOB                                (NAME)          (ACCOUNT NUMBER)
/ASSIGN  3=ICES
/ASSIGN  2=STRUDL
/RESTORE SYS1
STRUDL
PROBLEM 'EXAMPLE' 'DUAL METHOD'
UNITS  FEET POUNDS RADIANS
TYPE DUAL PLATE GENERAL NONSYMMETRIC
NODE COORDINATES
1 X 0. Y 2. B
2 X 1. Y 2. B
3 X 2. Y 2. B
4 X 2. Y 1. B
5 X 2. Y 0. B
6 X 1. Y 0. B
7 X 0. Y 0. B
8 X 0. Y 1. B
9 X 1. Y 1.
ELEMENT INCIDENCES
1 8 2 1
2 8 9 2
3 9 3 2
4 9 4 3
5 7 9 8
6 7 6 9
7 6 4 9
8 6 5 4
BOUNDARY INCIDENCES
'OUTER' 1
UNITS KIPS
ELEMENT PROPERTIES
1 TO 8 THICKNESS .25 DENSITY .15  TYPE 'FTOD' G 8000.
1 TO 4 EX 430000. EY 200000. PX .18
5 TO 8 EX 200000. PX .15
BOUNDARY CONDITION 'OUTER' RIGID FREE
XP 2.0 YP 2.5 FX .2  FY .3 MP .13
3 TO 1 CHI 0. EPSILON 0.
CLAMPED EDGE 'OUTER' 7 TO 5
BOUNDARY CONDITION DISPLACEMENT BENDING
1 TO 7 W .022 R 0.
3 TO 2 W 0. R 0.
UNITS POUNDS
LOADING 'SAMPLE1' 'EXAMPLE OF LOADING COMMANDS'
NODE LOADS
8 FORCE X 200. Y 300.
4 FORCE X 250.
```

```
LINE LOAD
7 FORCE X 100.
9 FORCE X 200.
3 FORCE X 100.
GRAVITY LOAD ANGLE-X 0. ANGLE-Z 1.10
LOADING 'SAMPLE2' 'EXAMPLE OF SPECIFIED PARTICULAR BEND. SOL.'
BENDING PARTICULAR SOLUTION
NODES 1 2 3 KX 0. KY .003
NODES 8 9 4 KX 0. KY .006
NODES 7 6 5 KX 0. KY .002
END
LOADING 'SAMPLE3' 'EXAMPLE OF STANDARD PARTICULAR SOLUTION'
BENDING PARTICULAR SOLUTION UNIFORM LOAD 100. C 0.5
LOADING LIST 'SAMPLE1' 'SAMPLE2'
FINITE ANALYSIS
FINISH
/END OF FILE
```